ANIMATION DE FLUIDE AVEC DES PARTICULES SUR UN MAILLAGE

par

Khalid Maina Abdoulaye Djado

Thèse présentée au Département d'informatique
en vue de l'obtention du grade de philosophiæ doctor (Ph D)

FACULTÉ DES SCIENCES

UNIVERSITÉ DE SHERBROOKE

Sherbrooke, Québec, Canada, 20 janvier 2011

Your file Votre référence
ISBN 978-0-494-75068-1
Our file Notre référence
ISBN 978-0-494-75068-1

Le 9 février 2011,

le jury a accepté la thèse de Monsieur Khalid Maina Abdoulaye Djado
dans sa version finale

Membres du jury

Professeur Richard Egli
Directeur de recherche
Département d'informatique

Professeur Jean-Pierre Dussault
Membre
Département Informatique

Professeur Benoît Crespin
Membre externe
Département d'informatique
Université de Limoges

Professeur Pierre-Marc Jodoin
Président rapporteur
Département d'informatique

Sommaire

Cette thèse porte sur l'animation d'un fluide à base de particules en utilisant un maillage dans un cadre d'infographie De nouvelles façons d'animer un fluide dans son espace de simulation sont explorées Cette animation se fait à travers la simulation, la visualisation d'un champ de vitesses et le rendu d'effets spéciaux de fluide

Il s'agit d'une thèse par articles dans lequel trois articles ont été réalisés Les deux premiers ont déja été publiés et le troisième est en cours de publication Le premier article porte sur une méthode de simplification de la dynamique des fluides par des précalculs du champ de vitesses sur un maillage cubique du domaine de simulation Plusieurs méthodes de visualisation sont proposées, y compris avec des particules Dans le second article, nous visualisons un champ de vitesses sur le maillage triangulaire d'une surface arbitraire Des particules sont introduites dans le champ de vitesses afin de le visualiser, mais aussi afin de créer des rendus de liquide ou de fumée sur la surface Le dernier article porte sur la simulation de gouttes d'eau sur une surface en temps réel La simulation de la condensation de l'eau sur une surface de même que la sueur ont pu être traités comme des applications

Sommaire

Remerciements

Je remercie très sincèrement mon directeur de recherche Richard Egli pour la confiance qu'il a eue en moi à plusieurs reprises Je te remercie pour m'avoir fait découvrir un domaine aussi passionnant que l'infographie Tu m'as appris l'infographie et tu m'as également appris à l'enseigner à d'autres personnes Merci encore !

Je remercie du fond de mon coeur ma famille pour m'avoir aidé durant toute ma vie Merci beaucoup à vous papa, maman, Oussou, Souley, Leila, Mimi, Maman-keyna, tonton Djoumassi, tantie Mehaou, tonton Elhadj keyna, etc pour votre amour et votre aide durant toutes mes études

Je remercie mes amis Kader, Maimouna, Ghislain, Omar, Serge, Alexandra, Nina et Brigitte pour votre soutien

Je remercie mes collègues et amis Eric, Gilles-Philippe, Bilel, Martin, Jean-François et Olivier pour votre aide et votre soutien

Je remercie le personnel de la Faculté des sciences et du Département d'informatique pour sa gentillesse et l'aide qu'il m'a apportée à chaque fois que j'en ai eu besoin Mention spéciale à Lise Charbonneau et Lynn Lebrun Merci aussi aux membres du MoIVRe de même que les membres du laboratoire d'optimisation

REMERCIEMENTS

Table des matières

TABLE DES MATIERES

Introduction

L'animation de fluide a suscité beaucoup d'intérêt en infographie il y a maintenant plus de deux décennies L'animation est composée de la partie simulation et de la visualisation du champ de vitesse créé La visualisation conduit au rendu de fluide La simulation du fluide se charge du calcul de la dynamique des fluides, notamment la résolution des équations de la dynamique des fluides comme Navier-Stokes par des méthodes numériques ou encore des heuristiques qui approximent ces équations Les méthodes présentées dans cette thèse utilisent aussi bien des méthodes numériques issues des équations de Navier-Stokes que des heuristiques pour simuler le fluide

Il existe généralement deux approches en simulation des fluides les méthodes eulériennes et les méthodes lagrangiennes [3] Une méthode eulérienne consiste à regarder, par rapport à un point fixe, comment les quantités physiques tels la densité ou la vitesse, changent par rapport au temps La méthode lagrangienne prend le fluide comme un ensemble de particules et les étudie afin de déterminer leur position et leur vitesse Cette thèse porte sur une approche de simulation par la méthode eulérienne La simulation s'effectue en utilisant un maillage de l'espace du fluide Nos travaux portent en premier sur un maillage cubique régulier et ensuite sur un maillage surfacique de triangles du domaine de simulation

La visualisation porte sur l'étude de la vitesse du fluide à chaque endroit du domaine de simulation Il s'agit aussi de déterminer la vitesse du fluide par des informations visuelles Il existe deux méthodes populaires de visualisation d'un champ de vitesses à savoir celle utilisant la déformation de texture et celle utilisant les mouvements de particules Dans cette thèse, les particules sont utilisées pour visualiser le champ de vitesses Ces particules sont ensuite utilisées pour rendre le fluide

Les travaux présentés dans cette thèse portent également à animer un fluide sur une

surface en utilisant des particules Les premiers travaux en simulation de fluide sur une surface en infographie ont été introduits par Stam [15] et Shi et Yu [14] Ces travaux utilisent des textures afin de visualiser le fluide D'autres travaux utilisant des textures pour la visualisation ont par la suite été publié [7, 11, 6] La visualisation par les textures limite les applications et les types de rendu que l'on peut obtenir Il existe quelques travaux sur la visualisation de champ de vitesses de fluide par des particules [17, 12, 2, 10] Ces travaux restent souvent peu efficaces dans le cas d'un champ de vitesses sur une surface pour des effets spéciaux de fluide Récemment Paillé [13] a publié une nouvelle méthode de simulation de fluide circulant à la surface d'un objet La méthode utilisée par Paillé est basée sur « smoothed particle hydrodynamics » Cette nouvelle méthode utilise une paramétrisation de la surface de certaines catégories d'objets Nous présentons de nouvelles façons de simuler, de visualiser et de rendre un fluide sur une surface Les particules permettent de créer des effets de fluides difficiles à reproduire avec les textures En effet, les particules permettent de simuler de la fumée ou de l'eau sur une surface Un modèle de visualisation d'un fluide sur une surface avec des particules ne suffit pas pour bien traiter de petites quantités de fluide de la taille d'une goutte d'eau Les gouttes d'eau ont des propriétés physiques différentes de celles de l'eau qui s'écoule en quantité importante Des travaux ont portés sur la simulation de gouttes d'eau sur une surface Certains de ces travaux comme [18] donnent des très bons résultats, mais portent sur des méthodes exigentes en temps de calcul D'autres travaux [9, 20, 19, 16, 1, 5] ne prévoient pas les cas où la goutte quitte la surface Nous présentons une nouvelle méthode de simulation de gouttes qui sont libres de quitter la surface en temps réel Nous avons pu ainsi simuler la condensation de l'eau sur une surface ou bien la sueur sur un visage humain en temps réel

Tout d'abord, nous avons commencé cette thèse par une étude de la dynamique des fluides suivant les équations de Navier-Stokes Il s'agit de calculer et de visualiser le champ de vitesses du fluide sur un maillage cubique de l'espace de simulation En raison de la complexité de résolution des équations de Navier-Stokes sur un maillage triangulaire arbitraire, les deux chapitres suivants utilisent des heuristiques qui décrivent bien le mouvement d'un fluide Le titre de cette thèse peut porter confusion car le premier article ne porte pas sur une simulation de fluide sur une surface Mais l'interaction avec la surface d'un objet présent dans le champ de vitesses a été tout de même étudiée

Dans le chapitre 1, nous avons élaboré une simplification d'un solveur des équations de

Navier-Stokes dans le but de pouvoir accélérer la dynamique des fluides pour une application interactive L'idée est de précalculer un champ de vitesses du fluide dans un maillage régulier 3D du domaine de simulation Une fois les étapes coûteuses en temps de calcul effectuées, nous utilisons ce champ de vitesses pour simuler le fluide par des interpolations Cette façon de faire est beaucoup plus rapide tout en permettant des résultats réalistes et efficaces dans un cadre de simulation interactive, pouvant être utilisée dans les jeux vidéo

Dans le chapitre 2, un champ de vitesses de fluide sur le maillage de la surface d'un objet a été simulé Nous nous sommes intéressés par la suite à la visualisation du champ de vitesses, puis à la production d'effets spéciaux de fluides comme l'eau et la fumée sur une surface Contrairement à tous les travaux qui ont abordé le même sujet, nous sommes les premiers à utiliser des particules pour produire le fluide Ceci ouvre une nouvelle façon de générer des effets de fluide sur des surfaces en utilisant des particules

Comme dernier article, nous nous sommes intéressés à la simulation et à la visualisation de l'écoulement de petites particules d'eau de la taille d'une goutte sur le maillage d'une surface et sous l'effet de la force gravitationnelle Le but principal de cet article est de pouvoir animer des gouttes d'eau sur une surface pour des applications interactives Une nouvelle méthode de modélisation de goutte d'eau est proposée, de même qu'une nouvelle approche pour simuler les comportements physiques comme la friction et la tension de la surface Comme principales applications, nous avons pu traiter le phénomène de condensation de l'eau sur des objets et la simulation de la sueur sur un visage humain en temps réel et de façon réaliste

Chapitre 1

Champ de vitesses 3D précalculées pour simuler la dynamique du fluide

Résumé

Cet article décrit une méthode de simulation 3D de fluide en utilisant un champ de vitesses précalculées Tout d'abord, nous présentons une méthode pour obtenir un champ de vitesses en utilisant la dynamique du fluide Le champ de vitesses du fluide est calculé sur une grille fixe dans le domaine du fluide Ensuite, nous présentons une méthode simplifiée de la dynamique du fluide en utilisant des vitesses précalculées et certaines heuristiques L'avantage d'utiliser un champ de vitesses précalculées est qu'il réduit de beaucoup le temps de calcul de la dynamique du fluide La plus grande partie du calcul de la dynamique du fluide se fait lors du processus de création du champ de vitesses et peut être précalculée

Commentaires

Ce premier article peut être considéré comme une mise en contexte sur la dynamique des fluides par rapport à la thèse globale Il vise un plus grand public tel que des développeurs en industrie, contrairement aux deux articles subséquents qui visent un public plus spécialisé en infographie Cet article constitue le chapitre 12 du livre *Game Engine Gems 1* de la maison d'édition *Jones and Bartlett*

Publishers édité par *Eric Lengyel* Cet article décrit les bases de la simulation des fluides avec Navier-Stokes en infographie, de même qu'une méthode simplifiée de la dynamique des fluides pour une application interactive ne nécessitant pas une grande complexité tout en visant des résultats réalistes Afin d'y arriver, nous utilisons des données précalculées de champ de vitesses permettant d'accroître la performance de la méthode proposée par rapport à l'utilisation classique des solutions aux équations de Navier-Stokes par des différences finies Le champ de vitesses précalculées est construit sur un maillage 3D cubique, régulier du domaine de simulation

La principale contribution de cet article est la méthode simplifiée de la simulation par différences finies utilisant Navier-Stokes Cette méthode présente de nombreux avantages dans un cadre d'animation interactive physique pour des jeux vidéo par exemple Une étude comparative du champ de vitesses simplifié et de celui provenant directement des équations de Navier-Stokes montre l'efficacité de la méthode proposée La méthode proposée est en moyenne deux fois plus rapide en FPS que le solveur de Navier-Stokes au complet

Cet article est partie d'une question simple à savoir qu'est-ce qui pourrait être simplifié dans la solution des équations de Navier-Stokes afin de réduire les temps de calcul ? Pour y arriver, nous commençons par implémenter un solveur de la dynamique des fluides bien connu en infographie décrit dans [8] Par la suite, nous avons simplifié ce solveur afin d'obtenir un champ de vitesses le plus proche possible de l'original Dans le cadre de cet article, j'ai proposé l'idée de la méthode simplifiée par les précalculs J'ai implémenté le code, validé la méthode proposée et rédigé l'article sous la supervision du professeur Richard Egli

Precomputed 3D Velocity Field for Simulating Fluid Dynamics

Khalid Djado
Richard Egli
Centre Moivre, Universite de Sherbrooke

Abstract

This gem describes a method for simulating 3D fluid dynamics by using a precomputed velocity field First, we present a method for building the fluid velocity field by using fluid dynamics The fluid velocity field is computed on a fixed grid in the fluid domain Second, we present a method for simplifying the fluid dynamics by using the precomputed velocities and some heuristics The advantage to using a precomputed velocity field is that it reduces the fluid dynamic computation time The greater part of the fluid dynamic computation is included in the velocity field computation process, which can be performed offline

1 Introduction

In recent years, much effort has been devoted to integrating real physics into the virtual world, in video games, for instance Some of these techniques, such as collision detection, are now very familiar to game developers and have been widely integrated into game physics engines This is not the case for fluids Simulating fluid dynamics well in virtual environments is generally a challenge Primarily, the difficulty is that there is no analytic solution for the Navier-Stokes equations used to describe the fluid dynamics In general, the computer graphics community uses a grid or particles to simulate a fluid This article uses an Eulerian 3D method on a grid to compute the fluid velocity field

One of the first studies on simulating fluids in computer graphics was done by N Foster and D Metaxas [2] Their work is based on a paper published by Harlow and Welch [3] A great summary of work on fluids can be found in a recent book [1] We have implemented the method of N Foster and D Metaxas [2] to compute the fluid dynamics, so the fluid domain is discretized into voxels The velocity field computation process is as follows

- The velocity source (or an exterior force) which we call the *blower* is placed inside the fluid domain (for example on a selected face of a voxel)
- The fluid velocity on the face of all voxels in the fluid domain is calculated by solving the Navier-Stokes equations These velocities can be stored in a file for future use, or precomputed before the simulation starts

The use of fluid dynamics in video games is very expensive in terms of computing time For performance gain, we precompute the physic of the fluid The key idea is to precompute the steps that take a long time Once the time-consuming steps in the calculation have been done, we use these data to simulate the fluid as if performing all calculations in real time The method is thus fast, while yielding realistic and acceptable results for interactive applications like video games The opportunities afforded by the approach used in this article are

- Computation of the fluid velocity anywhere in the fluid domain, using the precomputed velocities
- Simulation of a new velocity field, using the precomputed velocities and heuristics when the blower changes intensity and direction
- Simulation of the presence of a new object or the motion of an existing object in the fluid domain, using the precomputed velocities and heuristics This allows interaction between fluid and objects

2 Velocity Field Computation

The velocity field represents the fluid velocity anywhere in the simulation domain To obtain the velocity field, we start by computing the velocity on faces and then the velocity on voxels Each voxel has six faces The velocities on a voxel are shown by the red lines and the velocities of faces by the green lines in Figure 1 We use the finite difference method described in [2] to compute the fluid velocities on faces and voxels The blue line in Figure 1 represents the blower velocity introduced in the domain The Navier-Stokes equations are presented in Equation 1 and Equation 2

$$\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}+w\frac{\partial u}{\partial z}=-\frac{\partial P}{\partial x}+g_x+\nu\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}\right)$$

$$\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}+w\frac{\partial v}{\partial z}=-\frac{\partial P}{\partial y}+g_y+\nu\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}+\frac{\partial^2 v}{\partial z^2}\right) \qquad \text{Equation 1}$$

$$\frac{\partial w}{\partial t}+u\frac{\partial w}{\partial x}+v\frac{\partial w}{\partial y}+w\frac{\partial w}{\partial z}=-\frac{\partial P}{\partial z}+g_z+\nu\left(\frac{\partial^2 w}{\partial x^2}+\frac{\partial^2 w}{\partial y^2}+\frac{\partial^2 w}{\partial z^2}\right)$$

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}=0 \qquad \text{Equation 2}$$

Using Equation 1, Equation 2 and the product rule, we obtain the Equation 3

$$\frac{\partial u}{\partial t}+\frac{\partial u^2}{\partial x}+\frac{\partial uv}{\partial y}+\frac{\partial uw}{\partial z}=-\frac{\partial P}{\partial x}+g_x+\nu\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}\right)$$

$$\frac{\partial v}{\partial t}+\frac{\partial vu}{\partial x}+\frac{\partial v^2}{\partial y}+\frac{\partial vw}{\partial z}=-\frac{\partial P}{\partial y}+g_y+\nu\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}+\frac{\partial^2 v}{\partial z^2}\right)$$

Equation 3

$$\frac{\partial w}{\partial t}+\frac{\partial wu}{\partial x}+\frac{\partial wv}{\partial y}+\frac{\partial w^2}{\partial z}=-\frac{\partial P}{\partial z}+g_z+\nu\left(\frac{\partial^2 w}{\partial x^2}+\frac{\partial^2 w}{\partial y^2}+\frac{\partial^2 w}{\partial z^2}\right)$$

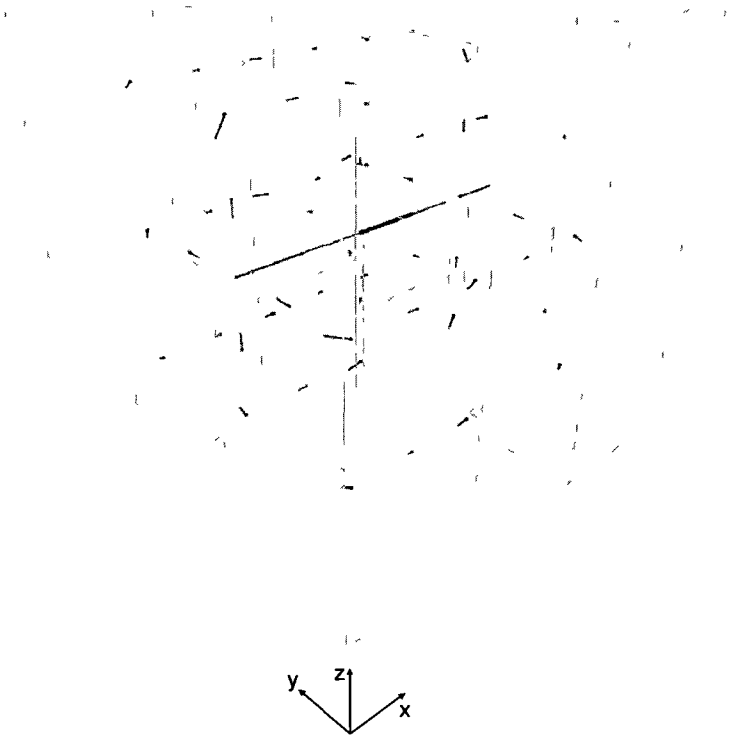


Figure 1

These equations were used in Foster [2] and for implementation in this paper
The quantities are summarized as follows

- u, v, w are the velocities of the fluid on a face or a voxel in the x, y and z axis directions, respectively
- P is the internal pressure of the fluid (formally, P is the pressure divided by the density of the fluid [4])
- ν is the viscosity coefficient
- g is the gravity force

Solving Equation 1 by the finite difference method gives the new velocity on each face of the fluid domain, like the velocities in green shown in Figure 1 These solutions can be found in [2] and in the source code provided with this article Note that the components of the velocity on a face along the 3 axes (x, y, z) are computed separately

Equation 2 is used to compute the null divergence of the fluid velocity This process leads to fluid velocity correction and, finally, the pressure update

To ensure the stability of the solver, the time step ∂t and velocities need to satisfy the condition $max\{u\ (\partial t/\partial x),\ v\ (\partial t/\partial y),\ w\ (\partial t/\partial z)\}<1$ 0 The steps of the solver are summarized in Listing 0 1

Listing 0 1 Pseudo-code of the full solver

```
1  Simulate(const float& deltaT)
2  {
3      // Reset Velocities (Boundary and Blower)
4      ResetBoundaryAndBlower(),
5      // Compute New Face Velocities on each voxel using Equation 3
6      for (int voxelID = 0, voxelID < m_voxelNumber, voxelID++)
7      {
8          UofFaceVelocity(voxelID, deltaT),
9          VofFaceVelocity(voxelID, deltaT),
10         WofFaceVelocity(voxelID, deltaT),
11     }
12     // Null Divergence step using Equation 2
13     float maxDivergence = s_epsilon,
14     while (maxDivergence >= s_epsilon)
15     {
16         float currentMaxDivergence = -INFINITE,
17         for (int voxelID = 0, voxelID < m_voxelNumber, voxelID++)
18         {
19             float divergence = 0 0F,
20             ComputeDeltaPressure(voxelID, deltaT, &divergence),
21             if (divergence >= currentMaxDivergence)
22             {
23                 currentMaxDivergence = divergence,
```

```
24            }
25        }
26        maxDivergence = currentMaxDivergence,
27        // Pressure Update
28        UpdatePressure(),
29    }
30    // Compute Voxel velocities by interpolation
31    ComputeVoxelVelocities(),
32 }
```

To simplify the fluid dynamics, we precompute the velocities of each face for a unitary blower in different directions such as x (See Figure 1) These precomputed velocities can be stored in memory before using the simplified simulation with heuristics They could also be saved on disk and then loaded into memory when needed

3 Physics simplification

Using the full fluid dynamics solver as presented in Section 2, we can change blower direction and move an object In this section we present heuristics that will allow us to simulate the presence and motion of an object, and also to change the blower velocity direction without using the full solver We have two kind of heuristics the "obstacle heuristic" and the "blower heuristic"

Obstacle heuristic

With full calculation, an obstacle like the black box shown in the Figure 2 requires us to recalculate all quantities (velocities and pressure) in the fluid domain at each time step The presence of an obstacle in the fluid domain entails a distortion in the velocity field We simplify the full solver by not computing lines 4 to 11 in Pseudo-code 1 and by setting the number of iterations for the null divergence step In fact, the divergence minimization process from lines 16 to 29 is in some cases performed more than 10 times when we set `s_fEpsilon=0 0001` In the case of the "obstacle heuristic" we set the number of iterations around 2 to make the process faster

The new velocity is computed by the interpolation described below Let V be the current velocity on a face, Vp and the precomputed velocity on the same face, c is a constant that is set to determine how fast the velocity field is returned to the precomputed state when an obstacle is removed ∂t is the time step The interpolation is done by Equation 4

$$V \leftarrow (1 - c \times \partial t) \times V + (c \times \partial t) \times V_p$$ Equation 4

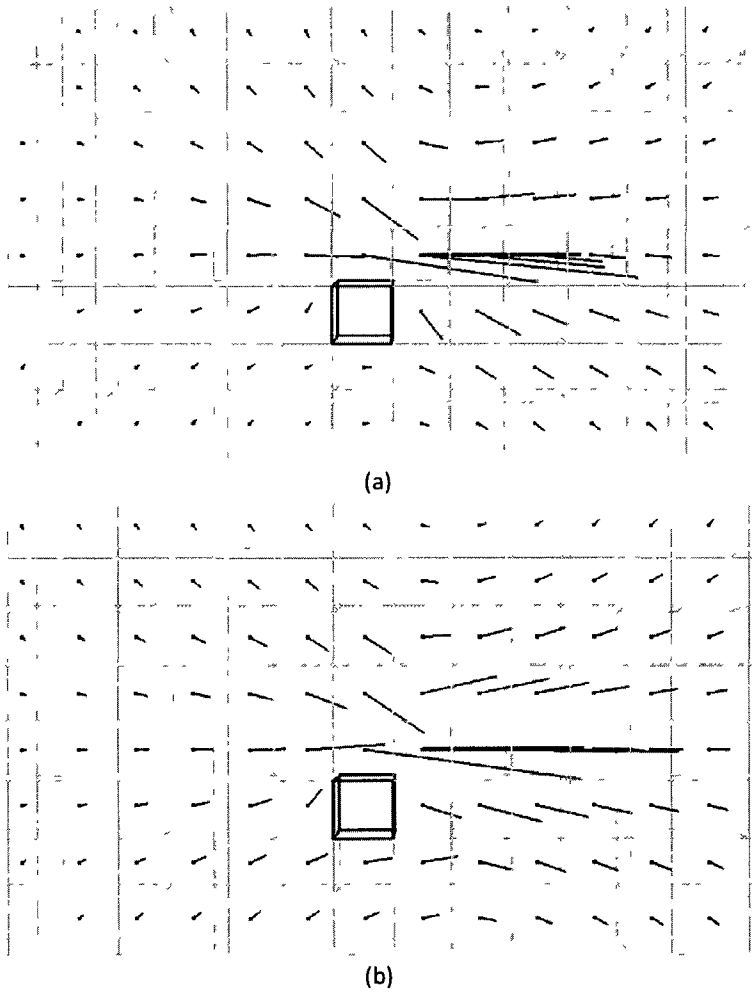


Figure 2 Images of the velocity field (a) from the full solver, (b) form the obstacle heuristic

The computation of a null divergence field allows velocity modification around the obstacle Null divergence means that the fluid flowing in equals the fluid flowing out In fact, the velocities on the faces of the obstacle are zero, so the null divergence ensures that the fluid on the neighboring voxels gets around the obstacle

The "obstacle heuristic" steps are summarized in Listing 0 2

Listing 0 2 Pseudo-code of the obstacle heuristic

```
1  SimulateObstacleHeuristic(const float& deltaT)
2  {
3     // Update Velocities using the Precomputation
4     // for each face of each voxel
5     FaceVelocity = (1 0F - cstObstacle * deltaT) * FaceVelocity
6        + (cstObstacle * deltaT) * PreCompFaceVelocity,
7
8     // Null divergence step in Precomputed Version
9     PreCompDivergence(deltaT),
10
11    // Compute Voxel velocities by averaging
12    ComputeVoxelVelocities(),
13 }
```

The main advantage of using the "obstacle heuristic" is that we are able to add and move objects in the fluid domain, starting from the precomputed velocity field without any obstacle The results obtained with the heuristic (see Figure 2-(b)) are similar to those of the full Navier-Stokes solver (see Figure 2-(a)), with the advantage that the process is at least twice as fast

Blower heuristic

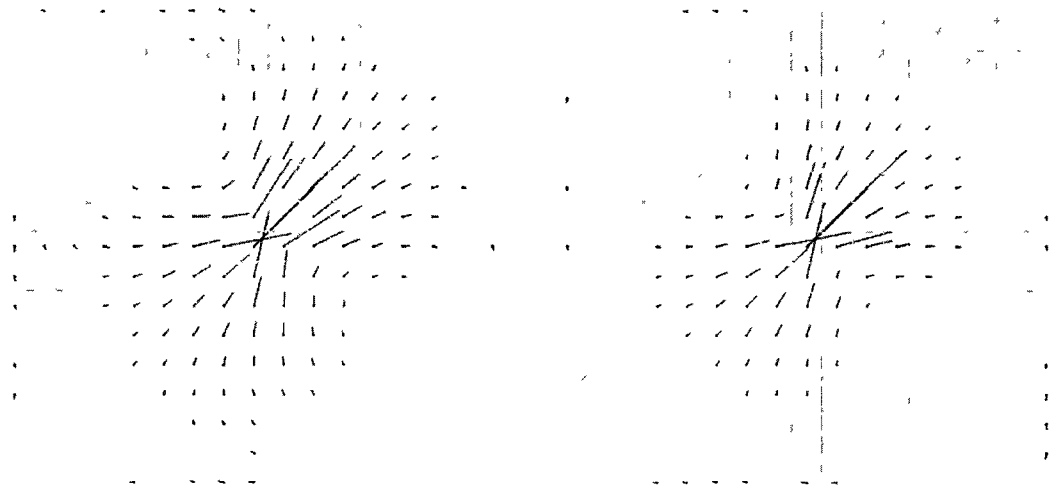
In the fluid simulation, any change in the blower velocity direction means all quantities (velocities and pressure) need to be recalculated in the fluid domain at each time step We simplify the full solver to be able to simulate a dynamic blower

By observing fluid simulation and changes in velocity, we notice that the velocity is linear with the norm of the blower velocity This means that if we double the velocity for the *blower*, the resultant velocities on the other voxels are also doubled We also notice that when the blower changes direction, each velocity of a voxel changes direction To be able to simulate the same effect, we use precomputed velocities by aiming the blower in each direction corresponding to the axes in the Cartesian coordinate system in the positive and negative directions In this article, we use blower directions in 2 dimensions along the x and y axes, but we think the method can be generalized to 3D We have to precompute the velocity field for the blower in directions (1 0, 0 0, 0 0), (-1 0, 0 0, 0 0), (0 0, 1 0, 0 0) and (0 0, -1 0, 0 0) Let θ be the angle between the new blower direction and the vector (1 0, 0 0, 0 0) For θ between $\theta 1$ and $\theta 2$ we proceed as follows

- We identify $\theta 1$ and $\theta 2$ according to θ For example $\theta 1 = 0$ and $\theta 2 = 90$ if θ is between 0 and 90 degrees
- We compute the interpolation weight $fFactor = (\theta - \theta 1)/(\theta 2 - \theta 1)$

- *Vp1* the precomputed face velocities for $\theta 1$
- *Vp2* the precomputed face velocities for $\theta 2$
- Each face velocity V is computed by $V = (1\ 0 - fFactor) * Vp1 + fFactor * Vp2$

This heuristic is an approximation of the velocities yielded by the full solver version Figure 3 shows a comparison of velocities from the "blower heuristic" and the full solver The simulation using the heuristic is more than twice as fast as the full solver



(a) (b)

Figure 3 Images of the velocity field (a) from the full solver, (b) form the blower heuristic

4 Results and Discussion

To visualize the fluid velocity field, the velocities on voxels can be displayed as vectors In this case each velocity is represented by a vector We also able to visualize the velocity field with unitary vectors for the direction and colors for the amplitude For example, the velocity can be depicted in blue for a high amplitude or red for a low amplitude It is also possible to visualize the velocity field with particles moving in the fluid domain The Figure 4 shows examples of visualization methods

To illustrate how the heuristics work, we provide an implementation with the article The program is written in C++ and uses OpenGL to display the 3D scene The user must set the resolution of the fluid domain in terms of number of voxels In the case of Figure 4, the fluid domain has 17x17x17 voxels and the blower (in blue) is at the voxel (8, 8, 8) We notice that the

grid in Figure 4 represents all voxels with z = 8 in the fluid domain In fact it is not easy to visualize the velocities with vectors when all 17x17x17 voxels are displayed

The full solver frame rate is around 172fps on a laptop equipped with an Intel CPU T2400 at 1 83GHz (dual core), with no parallelism in the simulation and visualization processes The same scene using "blower heuristic" (see Figure 4-(a)) allows a frame rate around 392fps We get 331fps with the two heuristics (see Figure 4-(b))

Some other optimizations are possible in a game physics context For example, we don't have to update the velocity field when the blower doesn't change and the obstacle doesn't move for a certain time The heuristics of this article can be simply added to an existing game physics engine The velocities can be precomputed at setup or loaded from a file

References

[1] R Bridson (2008) *Fluid Simulation for Computer Graphics*, A K Peters

[2] N Foster and D Metaxas (1996) "Realistic animation of liquids", *Graphical Models and Image Processing*, 58(5) 471–483

[3] Harlow, F H , and Welch, J E (1965) "Numerical calculation of time-dependent viscous incompressible flow," *Phys Fluids*, 8 2182–2189

[4] L Quartapelle *Numerical solution of the incompressible Navier-Stokes equations* Springer 1993

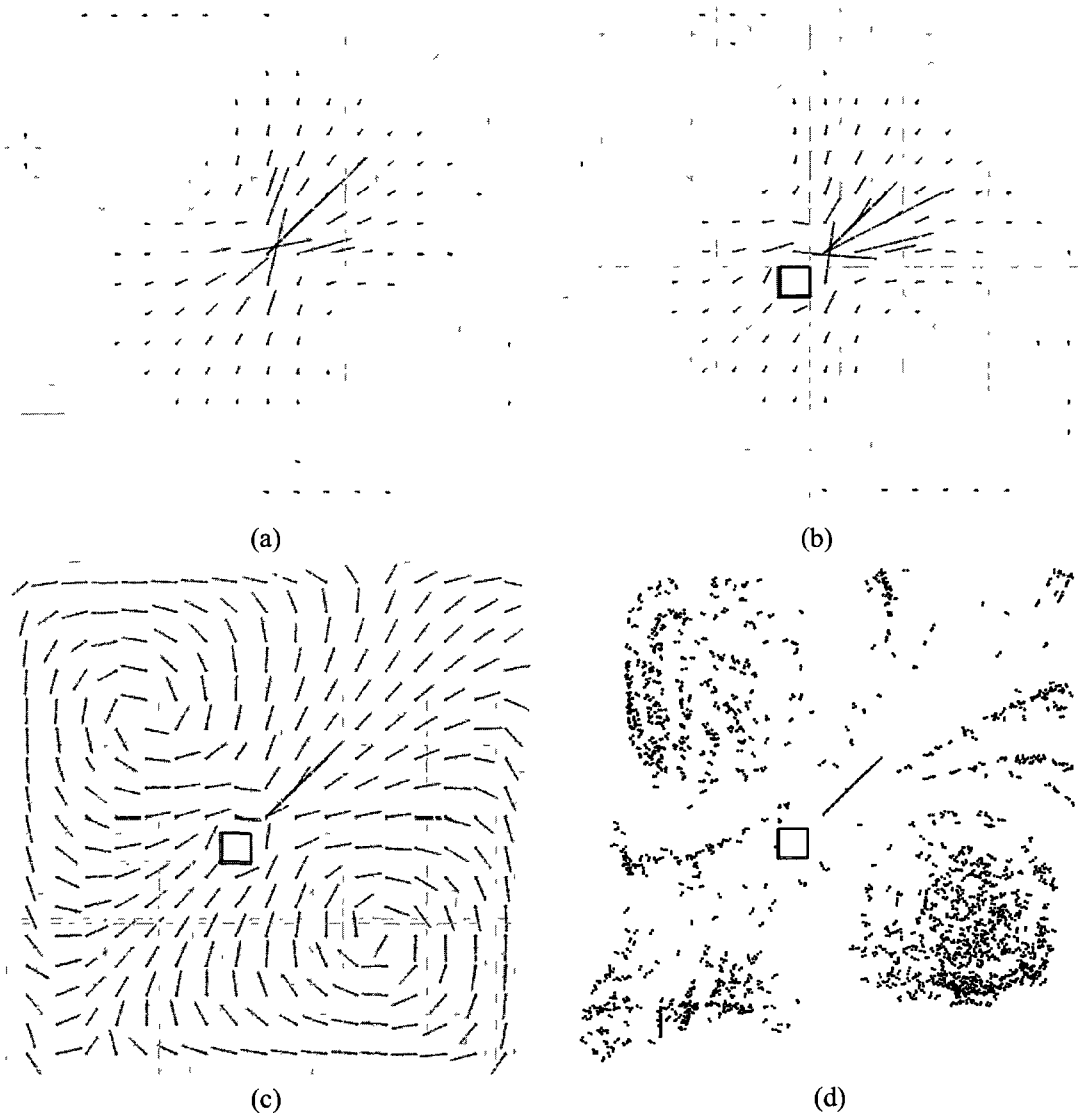


Figure 4 Images of the velocity field visualization using heuristics (a) with vectors without an obstacle, (b) with vectors with an obstacle, (c) with unity vectors for the direction and color for the amplitude with an obstacle, (d) with an obstacle and particles

Chapitre 2

Visualisation d'écoulement d'un fluide sur un maillage basée sur des particules

Résumé

Cet article présente une nouvelle méthode d'animation de particules sur un maillage arbitraire triangulaire à partir d'un champ de vitesses de l'écoulement d'un fluide La visualisation d'un champ de vitesses avec des particules permet de produire de nouveaux effets spéciaux Une contrainte d'attraction est mise en place pour corriger la vitesse d'une particule pour que sa trajectoire reste près de la surface Une contrainte de répulsion est également utilisée pour assurer une meilleure répartition des particules sur la surface Le champ de vitesses est calculé à l'aide d'heuristiques basées sur la physique mis en oeuvre via un modèle de chaînes Notre méthode est robuste et donne de bons résultats, en particulier pour les simulations interactives, même pour les systèmes avec plusieurs milliers de particules

Commentaires

Cet article porte sur la visualisation d'un champ de vitesses d'un fluide sur une surface arbitraire Contrairement au premier chapitre dans lequel le champ de vitesses est calculé dans une grille régulière 3D, ici le champ de vitesses est

calculé sur un maillage triangulaire d'une surface Une fois le champ de vitesses obtenu, nous y déposons des particules afin de créer des effets spéciaux de liquide ou de fumée sur la surface de l'objet La visualisation par plusieurs milliers de particules du champ de vitesses se fait en temps interactif, mais le rendu des effets spéciaux pour la fumée ou pour l'eau prend plusieurs dizaines de secondes par image Cet article a été présenté à *6th International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa* et publié dans l'acte de conférence

Dans cet article, nous voulons obtenir un champ de vitesses sur un maillage simulant la dynamique des fluides Pour obtenir le champ de vitesses, nous utilisons des heuristiques décrivant la dynamique des fluides sur un maillage triangulaire Ces heuristiques ont été introduites par Egli et Stewart [4] et utilisées comme applications dans un cadre de programmation unifié d'une méthode utilisant le modèle des chaînes Dans [4], le champ de vitesses est calculé uniquement sur une surface plane (en 2D) Nous avons modifié ces heuristiques pour une surface à topologie arbitraire Cet article est également une suite de la méthode 2D proposée dans [4]

Les contributions sont d'abord l'extension du générateur du champ de vitesses d'une surface 2D à une surface à topologie arbitraire Ensuite, il y a la génération d'effets spéciaux de fluide (avec la fumée et les liquides) sur une surface par une méthode de visualisation du champ de vitesses avec des particules Contrairement à tous les travaux ayant abordé la simulation de fluide sur une surface, la méthode que nous proposons est la première à utiliser les particules afin de visualiser le fluide Les résultats obtenus permettent de conclure que la résolution du maillage de la surface ne semble pas beaucoup influencer la visualisation du champ de vitesses Il est aussi important de noter que les distances géodésiques entre les sommets des triangles du maillage de la surface pourraient être pré-calculées pour la contrainte de répulsion entre les particules Mais il est possible que cela n'ait pas un impact majeur sur la qualité du résultat final L'idée de la simulation de fluide sur une surface avec des particules a été proposée par le professeur Richard Egli J'ai implémenté le code, validé la méthode proposée et rédigé l'article sous sa supervision

Particle-based Fluid Flow Visualization on Meshes

Khalid Djado * Richard Egli †

Centre MOIVRE, Departement d'Informatique
Universite de Sherbrooke, Sherbrooke, Québec, CANADA

Figure 1 Fluid on the Bunny s surface liquid and smoke

Abstract

This paper presents a new method for animating particles on an arbitrary triangular mesh starting from a fluid flow velocity field The velocity field visualization with particles allows the production of new special effects An attraction constraint is introduced to correct a particle s velocity so that its trajectory remains near the surface A repulsion constraint is also used to ensure a better distri bution of particles on the surface Our velocity field is obtained by physically-based heuristics implemented using a chain model Our method is robust and yields good results, in particular for interactive simulations, even for systems with several thousand particles

CR Categories I 3 7 [Computer Graphics] Three-Dimensional Graphics and Realism—Animation

Keywords fluid animation, flow on surface particles velocity field, visualization

1 Introduction

Fluid animation methods generally consist of two parts the simulation model that governs the movement of the fluid and the visualization of the created velocity field Simulation models fall into two broad categories those using Eulerian grids and those using Lagrangian particles For visualization, there are two popular approaches texture deformation or synthesis and the particle system In this paper, we visualize a fluid flow velocity field (created on a triangular grid by a Eulerian method) with particles

The first work on animation of a fluid on a surface was done by Stam [Stam 2003] followed by Shi and Yu [Shi and Yu 2004] The simulation model in [Stam 2003] is an extension of a 2D Navier Stokes fluid solver to curvilinear coordinates over Catmull-Clark surfaces, by using global surface parametrization In [Shi and Yu 2004], the simulation model is built directly on the mesh space for inviscid and incompressible flow Fan et al [Fan et al 2005] also tackled fluid simulation on a surface by using an unstructured Lattice Boltzmann Model (LBM) from 2D meshes to 3D surface meshes Lui et al [Lui et al 2005] present a method for solv ing Navier-Stokes equations on manifolds using a global conformal parametrization Recently Elcott et al [Elcott et al 2007] used an arbitrary simplicial mesh to define a fluid domain and simulate a fluid on a surface by using differential forms to solve the Navier-Stokes equations All of these methods use texture deformation or synthesis to visualize the velocity field on the surface

A common way to visualize a velocity field is by deforming or syn thesizing a texture under the effect of the field [van Wijk 2002, Weiskopf et al 2003, Li et al 2003, Laramee et al 2004, Weiskopf and Ertl 2004, Li et al 2008] Flow visualization with particles has been the subject of much work, including [Bauer et al 2002 Kruger et al 2005] Certain authors have considered 3D flow visualization near a surface [van Wijk 1993, Max et al 1994] These studies [Bauer et al 2002, Kruger et al 2005 van Wijk 1993, Max et al 1994] have been primarily interested in the interaction between particles and the surface of an object in the velocity field The object is an obstacle in the 3D velocity field The specific case of flow-on-surface visualization using particles has not been explored The principal contribution of this paper is to visualize fluid flow on surface with particles

Particles are commonly used in computer animation The main reason for using a particle system for velocity field visualization is to obtain other classes of special effects for fluid flow on a surface which are difficult to realize with texture deformation or synthesis (see Figure 1) Our visualization method with a particle system uses a simple velocity interpolation technique and an attraction constraint to keep the particles near the surface We also use a repulsion, as in [Witkin and Heckbert 1994] but for arbitrary meshes, to ensure a better distribution of particles on the surface

Like the studies described in [Shi and Yu 2004 Fan et al 2005, Elcott et al 2007] our method works directly on an arbitrary mesh Our velocity field is obtained by very simple physically-

*Khalid Djado@USherbrooke ca
†Richard Egli@USherbrooke ca

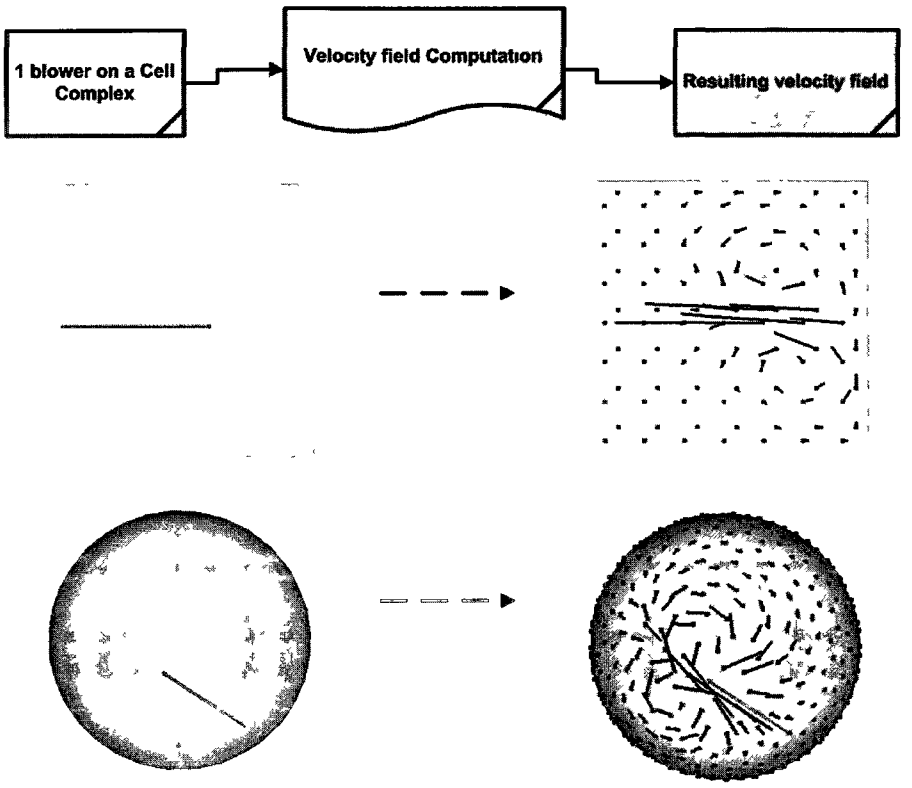


Figure 2 Summary of the velocity field computation Form left to right we first add a blower in red (fluid velocity source), then we compute the velocity field by simulation The result of the simulation is a 2-chain (velocities inside 2-cells)

based heuristics But the particle-based visualization method proposed in this paper can work with any velocity field on arbitrary meshes using other fluid simulation techniques The physically-based heuristics used here are an extension of a 2D fluid simulation work described in [Egli and Stewart 2004] We modify these heuristics for an arbitrary surface in 3D The 2D simulation method in [Egli and Stewart 2004] is written in a high-level topological and algebraic data structure we have called C3 (Chain and Cell Complexes) We use the same C3 formulation C3 is defined with the aim of finding neighbor and adjacency information easily In fact, the management of a transferred fluid mass from a triangle to its neighbors is very simple to write in the C3 formalism

In this paper we propose a new approach to animate a fluid on a surface with arbitrary topology Our specific contributions are

- a visualization of a flow on a surface using particles, in contrast *to all related work in computer animation [Stam 2003 Shi and Yu* 2004, Fan et al 2005, Lui et al 2005, Elcott et al 2007],
- an attraction constraint toward the surface and a repulsion between particles in the context of a flow on surface
- other classes of special effects such as volumetric smoke and liquid flows,

The rest of this paper is organized as follows in section 2, we give a basic overview of the C3 formulation, with some definitions and a summary of the velocity field computation process by simulation Section 3 describes the velocity field visualization method Then we give our results with some details on the implementation in section 4 Section 5 presents our conclusion

2 Fluid Velocity Field

2 1 Basic Overview of C3

In computer graphics, the implementation of a surface subdivision program like the Catmull Clark surface subdivision [Catmull and Clark 1978] or mesh simplification [Hoppe 1996] requires information on the neighbor and adjacency relationships between triangles edges and vertices The cell complex concept was developed as a way of finding neighbor and adjacency information easily A cell complex is the description of a system in terms of cells of different dimensions For example a surface mesh of a 3D object is a 2-complex and the 0-cells, 1-cells and 2-cells are respectively, the vertices edges and faces of the polygons (0-cell = vertex 1-cell = edge and 2-cell = face) The concept of chain is used to manipulate quantities on a cell complex

A p-chain is defined over a cell complex and assigns a coefficient to each p-cell in this cell complex For example the fluid mass on each 2-cell can be stored in a 2-chain of reals In certain applications coefficients may be associated with cells of different dimensions

A complete fluid animation on a plane based on heuristics and im-

plemented in C3 formulation is presented in [Egli 2000, Egli and Stewart 2000b, Egli and Stewart 2004] We use the same framework and its application programming interface (API) here More details about the C3 framework and its API can be found in [Egli and Stewart 2000b, Egli and Stewart 2000a, Egli 2000, Egli and Stewart 2004]

One of the major advantages of chains is their use for the description of models We first define and initialize all chains which can be used in the simulation model Formulas and laws are then applied directly to the chains This allows for global, unified treatment of the cell complex, simply as a transfer between cells inside the complex (fluid mass for example) The implementation of a fluid flow on an arbitrary surface becomes very simple by using the C3 API

2 2 Velocity Field Computation

The velocity field used in this paper is obtained by a fluid simulation on the cell complex At first we place flow sources (velocity sources) on selected 2-cells that we call the *blower* For example in the left image of Figure 2, we place only one blower, then we simulate the fluid after few time by using heuristics (four principles that we call the *Neighbour*, *Pressure*, *Friction* and *Border* in the Appendix) Figure 2 shows the steps of the velocity field computation process More details about these heuristics are presented in the Appendix section The result of the simulation is a 2-chain of the average velocity inside the cell complex

The goal of this paper is to visualize a fluid flow on surface with particles The input of our visualization method is a velocity field This velocity field can be obtained by the method described in the appendix or other fluid simulation method

3 Fluid Visualization Method

In flow-on-surface visualization with textures it is sufficient to know the velocity of the fluid anywhere on the surface An interpolation from the velocities on 0-cells allows one to obtain the fluid velocity anywhere inside a 2-cell In the case of particles it should be noted that they do not necessarily remain on the surface Without any attraction, we note that the particles will eventually get away from the surface This is more obvious when the particle s velocity is large and when the particle is located in places with high curvature of the surface We use an attraction-like constraint to keep particles near the surface Once we ensure that the particles remain near the surface we see that after some time the particles concentrate in certain areas like in area with low velocity field (see Figure 6(c)) To prevent this we use repulsion between particles

3 1 Particle Velocity Interpolation

Our velocity field is summarized by the average fluid velocity on each 2-cell on the surface (cell complex), stored in the 2-chain $ch2\_va$ But we need the velocities on the 0-cells for the interpolation of each particle's velocity These velocities on the 0-cells are computed in a 0-chain $ch0\_va$ We note that velocity averaging starting from $ch2\_va$ and moving to $ch0\_va$ is not a valid approach For example as shown in Figure 3(a) we obtain a velocity near zero by using $(\vec{V_a} + \vec{V_b})/2$ This problem is mentioned in [Shi and Yu 2004] The solution proposed in [Shi and Yu 2004] is to apply a local flattening, achieved by parameterizations Rather than this solution, we use a simpler approach by interpolating the velocity directions and velocity norms separately Although this is not justified mathematically, we obtain good results with this approach (see Figure 3(b)) Since we have the velocities on the 0-cells, they are used to compute the velocity of each particle by interpolation

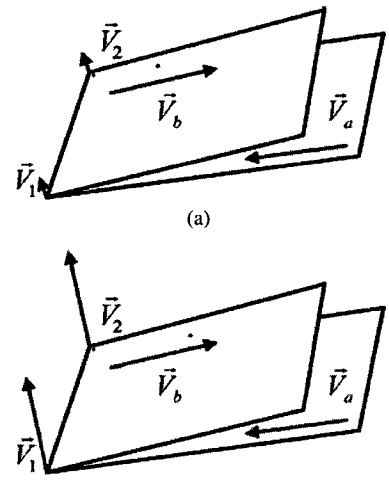


Figure 3 The velocity on a 0-cell (1 and 2) (a) by averaging velocities on 2-cells (a and b), (b) by averaging velocity directions and averaging velocity norms on the same 2-cells (a and b)

Let P be a particle position on the 2-cell ABC of Figure 4 The velocity $\vec{V_t}$ at point P at time t is determined by a quasi-barycentric velocity interpolation $\vec{V_A}$ $\vec{V_B}$ and $\vec{V_C}$ with weights $(S_A/(S_A + S_B + S_C))$ for vertex A $(S_B/(S_A + S_B + S_C))$ for vertex B and $(S_C/(S_A + S_B + S_C))$ for vertex C S_A, S_B and S_C are the areas of the three triangles of the tetrahedron PABC, as shown in Figure 4 We used quasi quasi-barycentric velocity interpolation because the projection of P on 2-cell ABC is not always inside of it, so barycentric couldn't be used Now, we just need to find the 2-cell associated with each particle

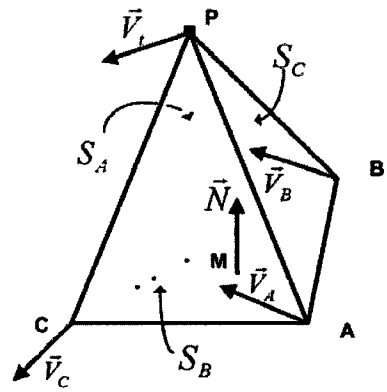


Figure 4 Velocity interpolation method for a particle at position P above the 2-cell composed of the A, B and C vertices

Each particle is associated with a 2-cell at each time step This localization of the nearest 2-cell to a particle is a problem similar to collision detection We must find the index of the 2-cell nearest to the particle at each time step To find this index, we use a Proximity

Query Package (PQP) [Gottschalk et al 1996, Larsen et al] The index returned by the PQP is updated in the particle attributes The velocity of the particle is interpolated from the velocities of 0-cells which constitute the returned 2-cell

3 2 Attraction Constraint

Let M be the middle point of the 2-cell shown in Figure 4 and $\overrightarrow{N}$ the normal vector of the 2-cell An attraction-like constraint is used to correct each particle's velocity Let K_a be the attraction factor For the particle P in Figure 4 the attraction is given by

$$\overrightarrow{A_t} = -K_a * (\overrightarrow{MP} \ \overrightarrow{N}) \ \overrightarrow{N} \qquad (1)$$

The distance between each particle and the 2-cell is given by $(\overrightarrow{MP} \ \overrightarrow{N})$ The particle velocity is changed by adding the attraction term $\overrightarrow{A_t}$ The $(\overrightarrow{MP} \ \overrightarrow{N})$ value changes sign depending on whether the particle is above or below the point M The new particle velocity is expressed by $\overrightarrow{V_t} \leftarrow \overrightarrow{V_t} + \overrightarrow{A_t}$

We allow the user to control the proximity of the particles to the surface via the constant parameter K_a and another constant parameter K_m K_m moves point M along the normal vector to the 2-cell In this way particles can be kept closer or farther to the surface

3 3 Repulsion Technique

For repulsion we use a Gaussian function of the type described in [Witkin and Heckbert 1994] The repulsion of particle i on particle j at respective positions P_i and P_j is expressed by

$$\overrightarrow{E_{ij}} = \alpha \ \exp(-\frac{\|\overrightarrow{P_iP_j}\|^2}{2\sigma^2}) \times \frac{\overrightarrow{P_iP_j}}{\|\overrightarrow{P_iP_j}\|} \qquad (2)$$

In Equation 2 α is an attenuation factor for the repulsion between particles and σ is the standard deviation making it possible to control the distribution of particles On the surface of a 3D object, it is more appropriate to use the geodesic distance rather than Euclidian distance to compute the repulsion between particles i and j But for reasons of performance and complexity, we use Euclidian distance between particles in a same area The obtained results are stable

An index of a 2-cell is associated with each particle Starting from this index, we use C3 to find 2-cells around a particle It is thus easy starting from a given particle i to find all particles j close to it (within a radius r) on the 2-cells around it The user specifies the size of the neighborhood by indicating the number of implied neighbors n and the radius r For example, n is 1 for the 2-cells neighbors to the current 2-cell n is 2 for the neighbors of the neighbors (including 2-cells for $n-1$) The repulsion depends on the distance of the nearby particles For each particle i the repulsion is expressed as a velocity correction $\overrightarrow{V_i} \leftarrow \overrightarrow{V_i} - \sum_{j \in S_i} \overrightarrow{E_{ij}}$, where S_i is a set of particles around i according to r and n In this way particles are better distributed on the surface

4 Results and Discussion

4 1 Summary of the Animation Method

Our complete flow method (simulation and visualization) is summarized by the diagram in Figure 5 Steps 1 to 3 are for the construction and initialization of C3 Steps 4 to 7 comprise the simulation process and return the average velocity on each 2-cell in the 2-chain $ch2\_va$ Steps 8 to 14 contain the particle-based velocity field visualization process Step 8 finds the fluid velocity for each 0-cell of the complex in the 0-chain $ch0\_va$ The velocity field is obtained starting from this 0-chain After step 14 we display the result particles move on the surface according to the velocity field

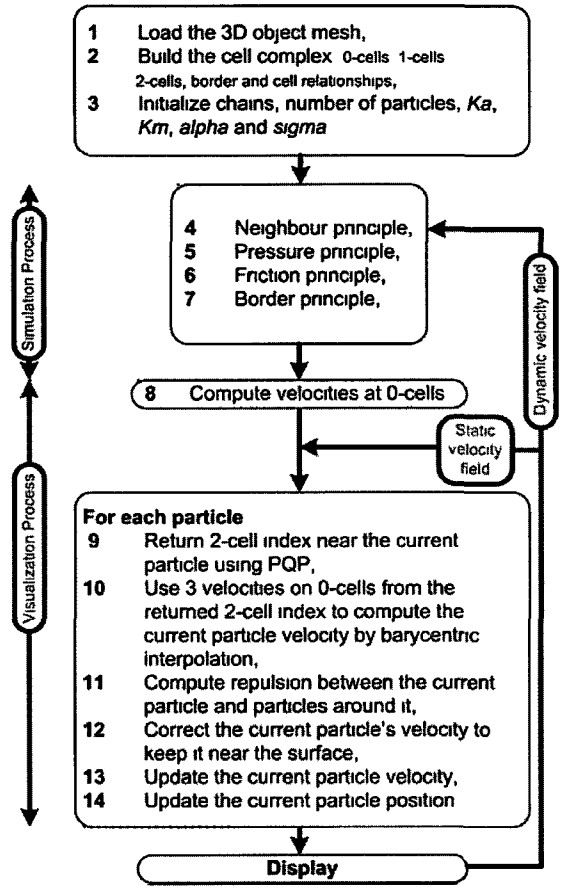


Figure 5 Summary of our flow model

We use two approaches in our flow method depending on whether the velocity field is static or dynamic In the static case, we fix the number of blowers, and we compute the velocity field during few time in order to allow the expansion of the fluid over the entire surface Then we carry out the velocity field visualization process

The static approach saves a lot of time, because the velocity field resulting from the simulation model is pre calculated in the 0-chain $ch0\_va$ In the dynamic case the blowers are dynamic so the steps from 4 to the display are repeated at each time step

4 2 Experimental Results

Our flow model has been tested on a sphere, a brezel model (from Javaview) and the Stanford bunny The code is implemented in C++, using OpenGL for the rendering, and our program runs on a laptop equipped with an ATI Mobility Radeon X1300 graphics card and an Intel CPU T2400 at 1 83GHz (dual core), with no parallelism in the simulation and the visualization process

As opposed to texture, particles offer the possibility of different effects such as smoke and liquid flows For instance, with particles it is possible to produce

- real-time fire or smoke by using sprites,
- liquids by using implicit surfaces
- volumetric smoke by using volumetric spheres

We present two examples of special effects produced by our flow visualization method a liquid and a volumetric smoke (using volumetric spheres) Some images resulting from animation of the tested models are given in Figure 1 Figure 6 illustrates the movement of the particles more clearly, as they are represented by small black spheres These results are for a static velocity field built after 400 time steps (where one time step = 0 005) The tested models contain 362 vertices (0-cells) and 720 triangles (2-cells) for the sphere 958 and 1920 for the brezel and 793 and 1561 for the bunny

To use our application, the users set blowers on selected 2-cells After the velocity field is computed particles are launched on the surface At $t = 0$, particles can be randomly placed on the surface as shown in Figure 6 (a), or placed on a selected 2-cell as in Figure 6 (f)

For the images in Figure 6 (f), (g) and (h), 500 particles are placed on one 2-cell on the bunny model This yields the volumetric smoke effects shown in right image of Figure 1 To obtain this effect we initialize $\alpha = 0\,0$ then set $\alpha = \alpha + 0\,005$ at each time step for a few seconds This avoids a strong repulsion of particles For the brezel model in Figure 6 (i) (j) and (k), 3000 particles are used

In Figure 6 (a) 3000 particles are randomly placed on the sphere with null velocity field Using repulsion alone, we obtain the image in Figure 6 (b) after a few frames In Figure 6 (c), we compute a velocity field with 3 blowers and no repulsion The particles are grouped together after a few seconds In Figure 6 (d) we add a slight repulsion and in Figure 6 (e) we use more repulsion

The frame rate for the sphere with 1000 particles is 31 fps For the brezel and the bunny with 1000 particles we obtain 22 fps and 32 fps, respectively For the same sphere with 3000 particles, we obtain 5 fps and 0 9 fps with 10000 particles In the case of a dynamic velocity field each blower has random direction and magnitude Thus the velocity field changes at each time step We apply a dynamic velocity field to the bunny with 1000 particles and the frame rate is 5 fps

We use data from particle positions to render liquid and volumetric smoke effects For the liquid, we use a blobby system to produce implicit surfaces and the results are produced by a ray tracing with reflective and refractive materials (see left image of Figure 1) The rendering time for the tested models varies from around 1 minute to 4 minutes per image For the smoke we use spheres with transparency, and noised and animated textures (right image of Figure 1) The rendering time is around 3 to 7 minutes per image The Figure 7 shows smoke and liquid effects on the sphere

The particle s velocity modified by the attraction and the repulsion constraints can cause a certain distortion of visualization compared to the methods of scientific visualization using textures, for example This distortion is more important when there is high repulsion between particles Our velocity field visualization method with particles is not necessarily intended for scientific purposes, but rather for special effects in computer animation The attraction and repulsion allow the user (an artist) to produce the effects he wants

5 Conclusion

In this paper, we have presented a new approach for animating particles on an arbitrary triangular mesh Unlike previous methods which simulate fluid flow on surfaces we use particles instead of texture to visualize the velocity field We add a repulsion between particles and an attraction on the surface Our method allows the production of new classes of special effects such as liquid and smoke flows Our approach uses no parametrization in either the simulation or the visualization process

Acknowledgements

This research was supported in part by grants from the Natural Sciences and Engineering Research Council of Canada The authors would like to thank Jean François Le Fillâtre Gilles-Philippe Paille Pierre-Marc Jodoin Lon Krung and Christian Preciado Castelo for their comments

References

BAUER D , PEIKERT, R , SATO M AND SICK M 2002 A case study in selective visualization of unsteady 3d flow In *VIS 02 Proceedings of the conference on Visualization 02* IEEE Computer Society, Washington DC, USA 525–528

CATMULL E , AND CLARK, J 1978 Recursively generated B-spline surfaces on arbitrary topological meshes *Computer Aided Design 10* (Sept), 350–355

EGLI R AND STEWART N F 2000 A framework for system specification using chains on cell complexes *Computer-Aided Design 32* 447–459

EGLI R , AND STEWART, N F 2000 Graphical simulation using chain models In *Technical Report 1174 Departement informatique et de recherche oprationnelle Universite de Montreal* July

EGLI R , AND STEWART N F 2004 Chain models in computer simulation *Math Comput Simul 66* 6 449–468

EGLI R 2000 Cadre de travail pour la specification de systemes avec des chaines sur un complexe cellulaire Phd thesis, Departement informatique et de recherche oprationnelle, Universite de Montreal, May

ELCOTT, S , TONG, Y , KANSO E , SCHRODER P , AND DESBRUN, M 2007 Stable, circulation-preserving, simplicial fluids *ACM Trans Graph 26*, 1 4

FAN, Z ZHAO, Y , KAUFMAN, A , AND HE Y 2005 Adapted unstructured lbm for flow simulation on curved surfaces In *SCA 05 Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 245–254

FOSTER N , AND METAXAS, D 1996 Realistic animation of liquids In *GI 96 Proceedings of the conference on Graphics interface 96* Canadian Information Processing Society Toronto Ont , Canada, Canada 204–212

GOTTSCHALK, S , LIN, M C , AND MANOCHA, D 1996 Obbtree a hierarchical structure for rapid interference detection In *SIGGRAPH 96 Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* 171–180

HOPPE, H 1996 Progressive meshes In *SIGGRAPH 96 Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 99–108

KRUGER, J , KIPFER, P , KONDRATIEVA, P , AND WESTERMANN, R 2005 A particle system for interactive visualization of 3d flows *IEEE Transactions on Visualization and Computer Graphics 11* 6 744–756

LARAMEE, R S , HAUSER, H DOLEISCH H , VROLIJK, B , POST F H , AND WEISKOPF D 2004 The state of the art in flow visualization Dense and texture-based techniques *Computer Graphics Forum 23* 2004

LARSEN, E , GOTTSCHALK, S LIN, M C , AND MANOCHA, D Fast proximity queries with swept sphere volumes In *Technical report TR99-018 Department of Computer Science University of N Carolina Chapel Hill*

LI, G -S , BORDOLOI, U D , AND SHEN, H -W 2003 Chameleon An interactive texture-based rendering framework for visualizing three-dimensional vector fields In *VIS '03 Proceedings of the 14th IEEE Visualization 2003 (VIS 03)*, 32

LI, G -S , TRICOCHE, X , WEISKOPF, D , AND HANSEN, C D 2008 Flow charts Visualization of vector fields on arbitrary surfaces *IEEE Transactions on Visualization and Computer Graphics 14*, 5, 1067–1080

LUI, L , WANG, Y , AND CHAN T F 2005 Solving pdes on manifold using global conformal parameterization In *Variational Geometric and Level Set Methods in Computer Vision Third International Workshop VLSM 2005* 307–319

MAX, N CRAWFIS R AND GRANT C 1994 Visualizing 3d velocity fields near contour surfaces In *VIS 94 Proceedings of the conference on Visualization 94*, IEEE Computer Society Press, Los Alamitos, CA USA 248–255

SHI, L AND YU, Y 2004 Inviscid and incompressible fluid simulation on triangle meshes *Comput Animat Virtual Worlds 15* 3-4 173–181

STAM, J 2003 Flows on surfaces of arbitrary topology In *SIGGRAPH 03 ACM SIGGRAPH 2003 Papers* 724–731

VAN WIJK, J J 1993 Flow visualization with surface particles *IEEE Comput Graph Appl 13*, 4, 18–24

VAN WIJK, J J 2002 Image based flow visualization In *SIGGRAPH 02 Proceedings of the 29th annual conference on Computer graphics and interactive techniques* 745–754

WEISKOPF, D , AND ERTL T 2004 A hybrid physical/device-space approach for spatio-temporally coherent interactive texture advection on curved surfaces In *GI 04 Proceedings of Graphics Interface 2004*, Canadian Human-Computer Communications Society School of Computer Science University of Waterloo, Waterloo, Ontario, Canada, 263–270

WEISKOPF, D , ERLEBACHER G , AND ERTL T 2003 A texture-based framework for spacetime-coherent visualization of time-dependent vector fields In *VIS 03 Proceedings of the 14th IEEE Visualization 2003 (VIS 03)*, 15

WITKIN A P , AND HECKBERT, P S 1994 Using particles to sample and control implicit surfaces In *SIGGRAPH 94 Proceedings of the 21st annual conference on Computer graphics and interactive techniques* 269–277

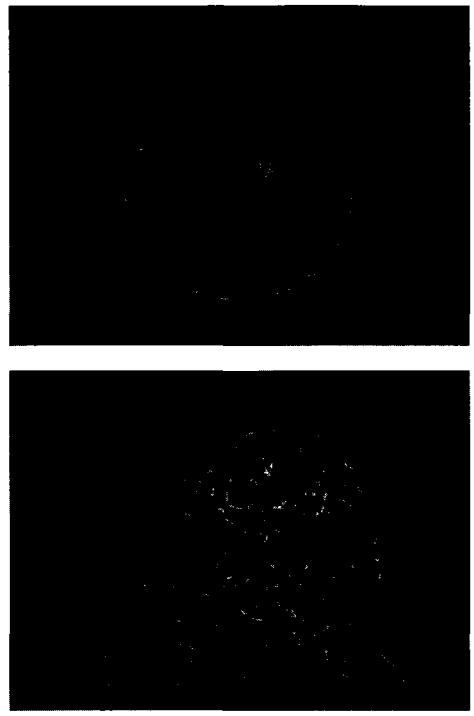


Figure 7 Smoke and liquid effects on the sphere

6 Appendix

The velocity field used in this paper is obtained by a fluid simulation on the cell complex by physically-based heuristics This physically-based heuristics, is an extension of a 2D fluid simulation work described by Egli and Stewart in [Egli and Stewart 2000b, Egli and Stewart 2004] The 2D results produced are visually comparable to those of more classical methods based on the Navier-Stokes equations [Foster and Metaxas 1996]

The simulation model is based on the model suggested by Egli and Stewart in which fluid advances on each 2-cell between time t and $t+dt$ according to four principles that we call the *Neighbour*, *Pressure*, *Friction* and *Border* principles

The quantities associated with each 2-cell at time t are the fluid mass m, the average velocity $\overrightarrow{v_a}$ and the density p ($p = m/s$ where s is the area of the 2-cell)

Neighbour principle The fluid mass in a 2-cell moves in the direction of the velocity $\overrightarrow{v_a}$, and part of it is transferred to neighbouring cells

Let m_{vout_i} be the outgoing fluid mass from the current 2 cell to its adjacent 2-cell i impelled by velocity $\overrightarrow{v_a}$ Let m_i be the initial fluid mass on the 2-cell i The fluid velocity $\overrightarrow{v_{ai}}$ on the 2-cell i is computed as follows

$$\overrightarrow{v_{ai}} \leftarrow \frac{m\ \overrightarrow{v_{ai}} + m_{vout_i}\overrightarrow{v_a}}{m_i + m_{vout}} \tag{3}$$

The fluid masses are updated by $m \leftarrow m - \sum_i m_{iout}$ and $m_i \leftarrow m_i + m_{vout_i}$

Pressure principle A part of the fluid mass in a 2-cell is displaced towards neighbouring 2-cells with lower pressure

Let m_{pout_i}, the outgoing fluid mass from the current 2-cell to its adjacent 2-cell i due to the pressure difference Let $\overrightarrow{v_{pi}}$ the velocity of the fluid mass acquired due to the pressure difference As in the *Neighbour* principle the fluid mass velocity $\overrightarrow{v_{ai}}$ on a 2-cell i is updated as follows

$$\overrightarrow{v_{ai}} \leftarrow \frac{m_i\overrightarrow{v_{ai}} + m_{pout_i}\overrightarrow{v_{pi}}}{m_i + m_{pout_i}} \tag{4}$$

The fluid velocity $\overrightarrow{v_a}$ on the current 2-cell changes according to

$$\overrightarrow{v_a} \leftarrow \frac{m\overrightarrow{v_a} + \sum_i m_{pout_i}\overrightarrow{v_{pi}}}{m_i + \sum_i m_{pout_i}} \tag{5}$$

As in the *Neighbour principle*, the fluid masses on the current 2-cell and its neighbour 2-cell i are updated by $m \leftarrow m - \sum_i m_{pout_i}$ and $m_i \leftarrow m_i + m_{pout_i}$

Friction principle There is friction between a 2-cell and its neighboring 2-cells, as well as internal friction within a 2-cell

We begin the calculations by using the updated values of the velocities obtained by application of the first two principles Due to the posited effect of internal friction, the velocity $\overrightarrow{v_a}$ of the fluid mass in a 2-cell decreases with time To model this phenomenon we use the following formula

$$\overrightarrow{v_a} \leftarrow (1 - k_f\Delta_t)\overrightarrow{v_a} \tag{6}$$

where k_f is a friction constant with $0 \leq k_f < \frac{1}{\Delta_t}$ (the value 0 corresponds to no friction) There is a similar effect related to inter-cellular friction

Border principle The direction of the average velocity $\overrightarrow{v_a}$ within a 2-cell tends to follow the *border* of the object

The *Border principle* leads to a heuristic to ensure that the directions of the average velocities $\overrightarrow{v_a}$ in each 2-cell touching the boundary of the geometric object will have a tendency to follow the boundary Every 1-cell that has only one adjacent 2-cell forms part of the boundary of the object The correction applied, in order to implement our *Border principle*, is this

$$\overrightarrow{v_a} \leftarrow (1 - k_b\Delta_t)\overrightarrow{v_a} + k_b\Delta_t\overrightarrow{v_b} \tag{7}$$

Here k_b is a constant that determines to what extent the velocity will have a tendency to follow the boundary (the higher the value of the constant, the larger the influence of the boundary on the velocity but the value of the constant must satisfy $0 \leq k_b\Delta_t \leq 1$), and $\overrightarrow{v_b}$ is the projection of the velocity $\overrightarrow{v_a}$ on the boundary

$$\overrightarrow{v_b} = (\overrightarrow{v_a} \times \frac{\overrightarrow{r}}{\|\overrightarrow{r}\|})\frac{\overrightarrow{r}}{\|\overrightarrow{r}\|}, \tag{8}$$

where $\overrightarrow{r}$ is the difference between the geometric positions of the 0-cells adjacent to the relevant 1-cell

The fluid simulation model using these four principles as described below, is only valid in the case of a planar triangular mesh A complete descriptions of these principles are given in [Egli 2000, Egli and Stewart 2000b, Egli and Stewart 2004]

Extension to a 3D Surface

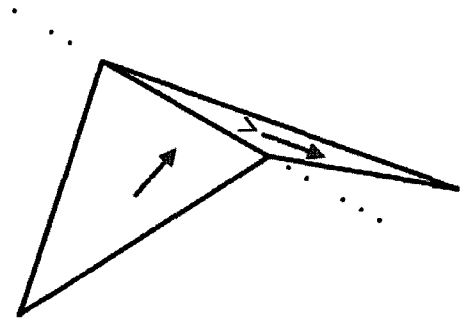


Figure 8 Transfer of fluid direction from a 2-cell to an adjacent 2-cell

To extend the model to surfaces with an arbitrary triangular mesh we mainly modify the *Neighbour* and *Pressure* principles In the arbitrary surface case (mesh in 3D), the vectors $m_{vout_i}\overrightarrow{v_a}$ in Equation 3 and $m_{pout_i}\overrightarrow{v_{pi}}$ in Equation 5 in the *Neighbour* and *Pressure* principles must be transferred to the appropriate planes (2-cells are not necessarily in the same plane) To address this problem, we add a rotation operator in the C3 API to make sure that the vectors are brought back onto the plane The Figure 8 illustrate the rotation of a vector from a 2-cell to an adjacent 2-cell

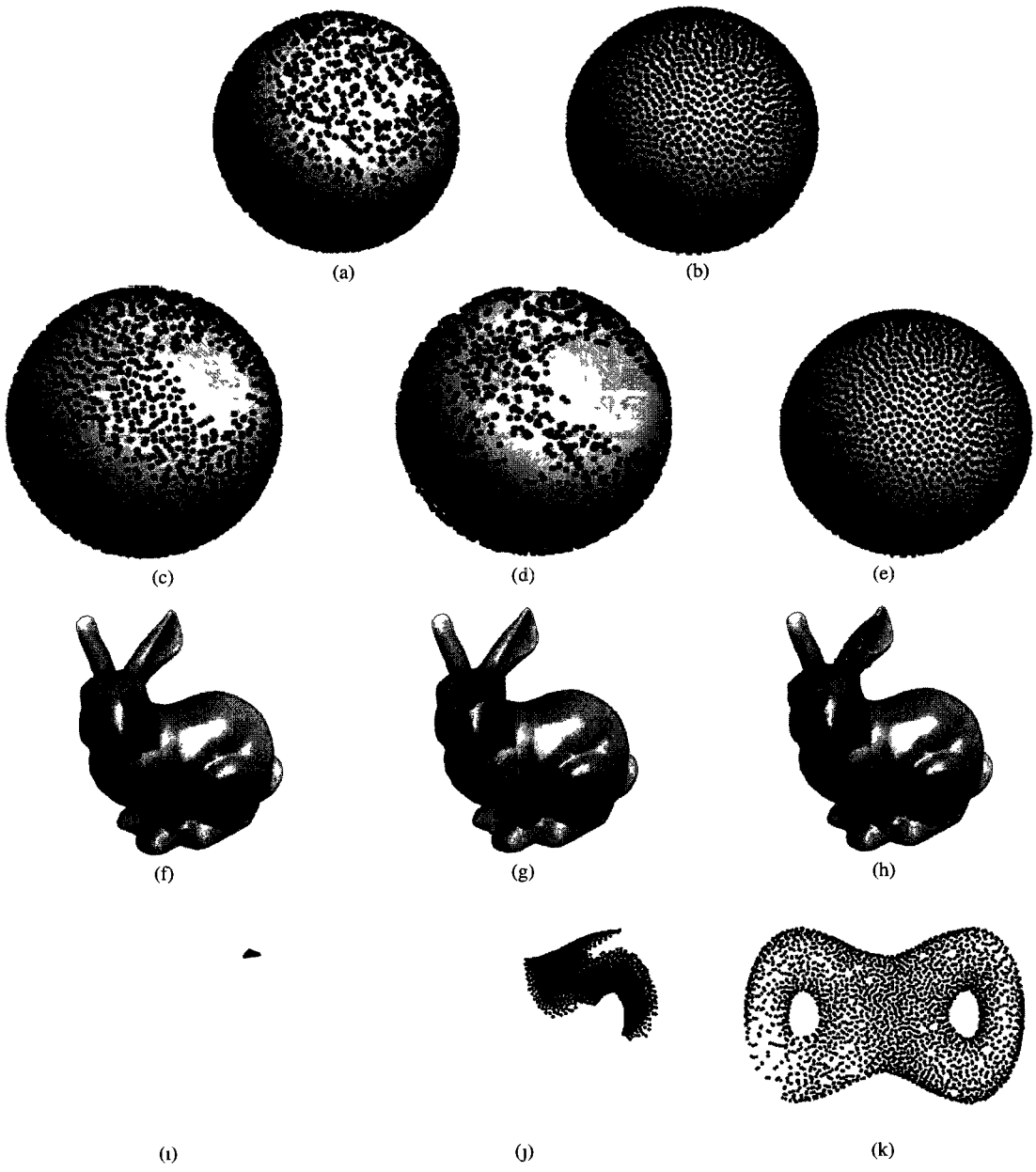


Figure 6 Illustration of the repulsion between 3000 particles on a sphere (a) null velocity field + null repulsion, (b) null velocity field + repulsion, (c) velocity field + null repulsion, (d) velocity field + slight repulsion, (e) velocity field + more repulsion 500 particles in a velocity field on the bunny model with repulsion (f) frame 1 (g) frame 50 (h) after frame 250 3000 particles in a velocity field on the brezel model with repulsion (i) frame 1, (j) frame 50, (k) after frame 250

Chapitre 3

Animation de gouttes basée sur des particules sur des maillages en temps réel

Résumé

Cet article présente une méthode pour simuler le mouvement des gouttes d'eau sur une surface en temps réel Nous décrivons la dynamique d'une goutte sur une surface, puis nous présentons notre modèle de simulation Nous utilisons une représentation géométrique pour la goutte Chaque goutte est modélisée par un maillage d'une surface 3D déformable Cette représentation géométrique permet à la goutte de rester sur la surface ou dans les airs Nous proposons aussi une méthode simple pour faire la fusion et la séparation de gouttes Pour le rendu, nous simulons la réflexion et la réfraction La trace d'eau laissée par les gouttes est également prise en compte dans la simulation Notre méthode est rapide, robuste, et permet d'obtenir des résultats réalistes notamment les applications pour traiter la condensation sur une surface ou la transpiration de l'homme en temps réel

Commentaires

Cet article porte sur la simulation de particules d'eau de la taille d'une goutte sur le maillage d'une surface Comme principale application, nous nous sommes intéressés à la condensation de l'eau sur des objets et à la simulation de sueur en temps réel Les gouttes sont entraînées par le champ gravitationnel

Tout comme au chapitre 2 nous travaillons sur un maillage triangulaire d'une surface arbitraire Contrairement aux chapitres précédents, nous n'effectuons pas de calcul de vitesses sur le maillage avant l'introduction des gouttes d'eau dans le champ de vitesses La vitesse de chaque goutte est calculée dynamiquement en fonction des forces qui s'exercent sur elle Ces forces dépendent de l'endroit où se situe la goutte sur la surface Dans cet article, toutes les étapes, de la simulation au rendu, se font en temps réel Cet article sera soumis au journal *Visual Computer*

La principale contribution de cet article est la bonne performance des applications produites avec le modèle proposé Contrairement à tous les travaux qui ont abordé le sujet, notre méthode est la seule permettant à une goutte de se déplacer sur une surface et de pouvoir la quitter en temps réel Le modèle de simulation proposé est simple à implémenter et permet d'avoir un mouvement de goutte réaliste sur une surface Ceci permet de rehausser le réalisme, tant du point de vue simulation que du point de vue rendu, dans des applications interactives nécessitant des gouttes d'eau Il faut noter que les méthodes de fusion et de séparation proposées dans ce chapitre ne se font pas de manière transitoire Dans la réalité, la fusion et la séparation des gouttes sont presque instantanées Lorsqu'une goutte d'eau quitte la surface, elle prend une forme sphérique L'idée de simuler des gouttes d'eau sur une surface est partie d'un projet de simulation réaliste de sueur dans les jeux vidéo Le projet m'a été suggéré par Fabrice Granger lorsque j'étais un de ses employés J'ai implémenté le code, validé la méthode proposée et rédigé l'article sous la supervision du professeur Richard Egli

Particle-based Drop Animation on Meshes in Real Time

Khalid Djado [1] * Richard Egli [1,†] Fabrice Granger [2,‡]
[1]Centre Moivre, Universite de Sherbrooke, Sherbrooke, Quebec, Canada
[2]Amusement Cyanide Inc , Montreal, Quebec, Canada

Figure 1 *Application to simulate water condensation at left image and the human face sweating at right image*

Abstract

This paper presents a method for simulating the motion of water drops on a surface in real time We describe the dynamics of a drop moving on the surface and then we present our simulation model We use a geometry-based representation of a drop Each drop is modeled by a deformable 3D mesh This geometrical representation allows drops to be on the surface or in the air We also propose a simple method to handle drop merging and separation For the rendering we simulate reflection and refraction The drop trace is also taken into account Our method is fast and robust and yields realistic results when applied to treat condensation on a surface or human sweating in real time

CR Categories I 3 7 [Computer Graphics] Three-Dimensional Graphics and Realism—Animation

Keywords particles, water drop, drop deformation drop trace human sweating, water condensation, liquid-solid interaction

1 Introduction

Today, fluids play an important role in physics-based animation research Much work has been done on fluids in recent years and some of these studies have been devoted to real-time applications The flow of water drops represents a challenge given the complexity of establishing a simple physical model to describe the phenomenon well In this paper we propose a simulation model for drop motion on a surface of arbitrary topology in real time

*Khalid Djado@usherbrooke ca
†Richard Egli@usherbrooke ca
‡fgranger@cyanide studio ca

Physically the surface of a water drop in the air is an interface between two fluids the water and the air When the drop is in contact with a solid surface, this surface must be taken into account in modeling its shape This is the principle of the triple contact line Using the complete triple contact line model for the animation of drops flowing over a surface is a very complicated problem It entails taking Young s relation into account [de Gennes 1985] solving the fluid dynamics, building and animating the drop shape taking the surface tension into consideration, performing drop merges and separations considering the properties of the fluid and the surface, etc To address this challenge, there is no physics-based model for drop simulation in computer animation except the paper of Wang et al [Wang et al 2005] The simulation model presented in [Wang et al 2005] uses a level set representation, Navier-Stokes equations, Young s relation and surface tension to create the liquid motion on the surface The results obtained are very realistic but the simulation takes several minutes per frame In this paper an empirical method is used for modeling the drop shape and a physics-based method governs the motion of drops

The method presented here allows realistic animation of drops on the surface The gravitational force and the surface tension are used, with emphasis on natural phenomena such as water condensation or human sweating, as shown in Figure 1 The drop is modeled by a dynamic 3D surface, starting from a sphere The mesh of the drop is deformed according to forces The mesh representation allows each drop to be on the surface or in the air, which is difficult to do with a vector normal perturbation or displacement map method We also propose a merging method based on the detection of collisions between drops, and suppose that the merging process between two drops takes place in a very short period of time A drop is also allowed to separate into two drops when a large force is exerted on it

The rest of this paper is organized as follows in Section 2, we

provide an overview of related work Section 3 describes our simulation model, including the drop shape deformation model the merging model and the separation model Experimental results and details on the implementation are given in Section 4 In Section 5, we present our conclusion and some directions for future work

2 Related Work

Human sweating and condensation are natural phenomena involving small quantities of liquid, the size of a water drop, on a surface In computer graphics, much work has been done on fluid simulation on surfaces Some recent studies using texture deformation are reported in [Stam 2003] [Shi and Yu 2004] [Fan et al 2005] [Lui et al 2005], [Elcott et al 2007] These papers do not address the situation of small quantities of liquid on a surface In [Djado and Egli 2009], particles are used to simulate liquid on a surface Their approach is able to treat a small quantity of water on a surface, but there is no interaction between water particles such as merging or separation Also, the water particles only follow an artificial velocity field on the surface

Some work has been done specifically on liquid drop simulation in computer graphics The most recent studies are reported in [El Hajjar et al 2009] and [Jung and Behr 2009] According to [El Hajjar et al 2009], work on drop simulation can be classified in two categories studies that place more emphasis on the shape of the drops [Fournier et al 1998] [Yu et al 1998] [Murta and Miller 1999], [Yu et al 1999], [Tong et al 2002] and those that focus primarily on the animation of the droplets on the surface [Kaneda et al 1993], [Kaneda et al 1996] [Kaneda et al 1999] [Iglesias et al 2001], [Sato et al 2002] [Yang et al 2004], [Takenaka et al 2008], [El Hajjar et al 2009], [Jung and Behr 2009] Work in this area can also be classified into studies applicable to 3D surfaces in general [Dorsey et al 1996], [Kaneda et al 1996], [Kaneda et al 1999] [Fournier et al 1998] [Murta and Miller 1999] [Iglesias et al 2001], [Sato et al 2002] [Jung and Behr 2009] and those confined to a 2D surface [Kaneda et al 1993] [Yu et al 1998] [Yu et al 1999] [Yang et al 2004], [Takenaka et al 2008] [Algan et al 2008] [El Hajjar et al 2009] The method presented in this paper focuses on drop shape animation on a 3D surface

In [Dorsey et al 1996], particles are introduced to simulate drops on a surface The drop shape is static because it is supposed to be small In [Kaneda et al 1996] and [Kaneda et al 1999] spheres are used to represent drops rendered by ray casting In [Fournier et al 1998], a drop is simulated using a mass-spring network Surface tension and volume conservation are also taken into account However the merging and separation processes become more difficult to do well In [Murta and Miller 1999] an adaptive particle system and an implicit surface are used to create the fluid The method described in [Iglesias et al 2001] simulates drops on the parametric surface The shape of the drop is static and the transition between patches may cause numerical instability In [Sato et al 2002] a GPU implementation of [Kaneda et al 1993] is presented The shape of the drop is still static the drop is a static hemisphere In [Jung and Behr 2009] particles and texture information in the parametric domain are used to simulate drops on the surface as a bump map It becomes difficult to build the shape of a drop when it moves away from the surface Here we present a new model to create the deformable shape of a drop and a realistic motion of this drop on a 3D surface In this paper we propose a new approach for animating a drop on a 3D surface with arbitrary topology in real time Our specific contributions are

- a deformable mesh for modeling the drop shape,
- a separation model and a merging model based on a collision detection method
- volume conservation in the simulation
- interesting and realistic applications like human sweating and water condensation on surfaces in real time, as shown in Figure 1

3 Simulation Model

From the microscopic point of view, the formation of a water drop is due to interactions between molecules These interactions in the liquid are called the surface tension The surface tension and surface properties create friction as the drop moves on a surface The drop may also sometimes remain glued to the surface as a result of surface tension

A water drop on a surface is subjected to three surface tensions at the triple contact lines This is characterized in equilibrium by Young's relation described in Equation 1 The surface tensions at the triple contact lines are the Solid-Liquid tension, indicated by λ_{SL} the Liquid-Gas tension, λ_{LG} and the Solid-Gas tension, λ_{SG} The angle between the Liquid-Gas interface and the Solid-Gas interface is denoted by θ_E

$$\lambda_{SG} - (\lambda_{SL} + \lambda_{LG} \cos\theta_E) = 0 \quad (1)$$

The equilibrium from Equation 1 is illustrated in Figure 2 More details can be found in [El Hajjar 2008] To simulate this physical phenomenon we simplify the drop shape modeling by using the motion of a virtual point we call the **C**entral **P**oint of **D**eformation (CPD) The motion of this single point is caused by the gravitational force This simplification allows us to get a realistic shape of a drop in real-time However, it is more difficult to solve Equation 1 when modeling thousands of drop shapes on a surface in real time

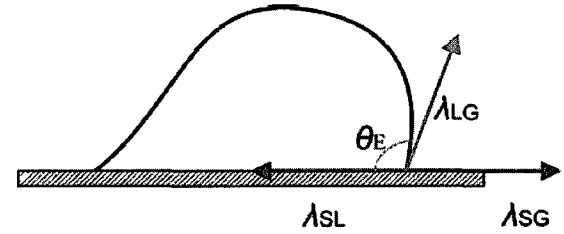


Figure 2 *Surface tensions at the triple contact lines Solid Liquid Liquid Gas and Solid-Gas*

To simulate the physical behavior of a water drop we start by defining the CPD and its motion due to gravity This deformation point is used in trigonometric functions to build the shape of the drop We allow drops to merge and separate Forces are computed at the end of this section

3 1 Central Point of Deformation

When a drop is on a horizontal plane we suppose that its shape is a hemisphere, as shown in Figure 3-(a) We suppose this hemisphere to be unitary, i e its radius r is 1 0 The center of mass of the unitary hemisphere, point G in Figure 3-(a), is at a distance of $\frac{3}{8}r$ from the hemisphere center

The CPD is the point of reference that is used to model the drop shape In general the center of mass of an object is computed by its shape The CPD is an approximation of the center of mass The CPD is the point G (resp G') in Figure 3-(a) (resp Figure 3-(b))

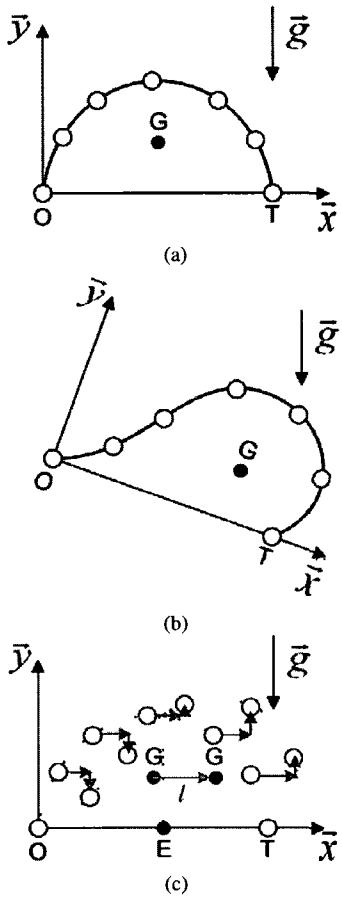


Figure 3 *Drop shape deformation model a) before the deformation (CPD at position G) b) after the deformation (CPD at position G) c) before and after the deformation illustration on the same image*

The gravitational acceleration $\vec{g}$ is indicated in Figure 3 The local orthonormal coordonate system is $(O, \vec{x}, \vec{y}, \vec{z})$

When the drop is on a slope relative to the horizontal, the drop is deformed as shown in Figure 3-(b) A lobe is created, according to the direction of $\vec{g}$ Let l be the length along the $x-$axis between CPD positions from the horizontal to the slope The length l is obtained by using the mass of the drop and the angle θ between the surface normal and $\vec{g}$ Its relation is given in Equation 2 The parameter m is the mass of the drop We define M_{min} to be the minimum mass required for a drop to start moving on the surface The parameter M_{max} is the maximum mass for a drop in the simulation The parameter λ is a constant for controlling the sensitivity of the drop deformation

$$l = min(\lambda \frac{m - M_{min}}{M_{max} - M_{min}}, 1\ 0) \sin\theta \qquad (2)$$

3 2 Drop Shape Deformation

We use trigonometric functions and l from Equation 2 to simulate the drop shape deformation This deformation is done by dynamically modifying the vertex positions of the hemisphere The small white circles in Figure 3 represent vertices of the mesh of the drop Figure 3-(c) shows the drop before (Figure 3-(a)) and after (Figure 3-(b)) the deformation on the same image The points O and T are respectively the head and the tail contact points between the drop and the surface along the $x-$axis The point E is the midpoint between O and T

The red (resp blue) arrows in Figure 3-(c) represent the vertex position changes along the $x-$axis (resp $y-$axis) between before and after the deformation Let f and g be the functions representing the graph along the $x-$axis and the $y-$axis, respectively The function f is graphed in Figure 4-(a) Figure 4-(b) shows the graph of function g

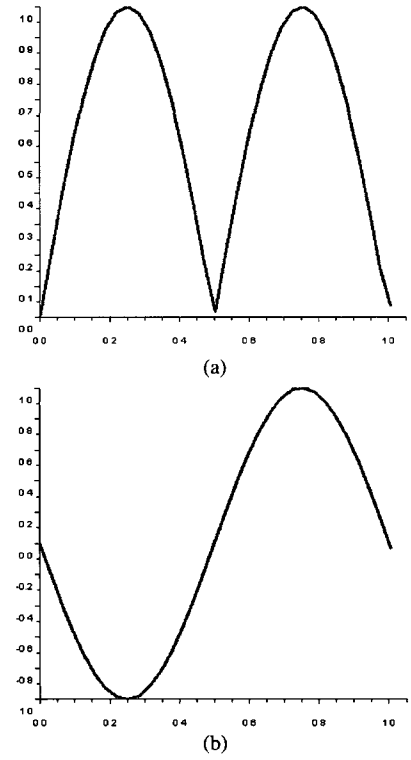


Figure 4 *Deformation functions f and g in the local coordonate system a) along the x axis b) along the y axis*

The functions f and g can be approximated by the trigonometric functions in Equation 3 and Equation 4 The variable x represents the coordonate of a vertex with coordonates (x, y, z) on the drop mesh along the $x-$axis in the local orthonormal coordonate system $(O, \vec{x}, \vec{y}, \vec{z})$

Each vertex with coordonates (x y z) in $(O, \vec{x}, \vec{y}, \vec{z})$ on the drop mesh changes its position according to the function h in Equation 5 We define two positive constants, C_1 and C_2, to allow the user to control the shape of the deformed drop By modifying the parameters λ from Equation 2 and C_1 C_2 from Equation 5, it is possible to simulate the viscosity of the fluid

The function h of Equation 5 is applied to an undeformed drop as shown in Figure 5-(a) When $l = 0\ 5$, the result obtained is the drop illustrated in Figure 5-(b) Figure 5-(c) shows the result when

$l = 1\,0$

$$f(x) = |\sin(2\pi x)| \tag{3}$$

$$g(x) = -\sin(2\pi x) \tag{4}$$

$$h(x,y,z) = (x,y,z)^T + (lC_1 f(x))\ (1,0,0)^T + (lC_2 g(x))\ (0,1,0)^T \tag{5}$$

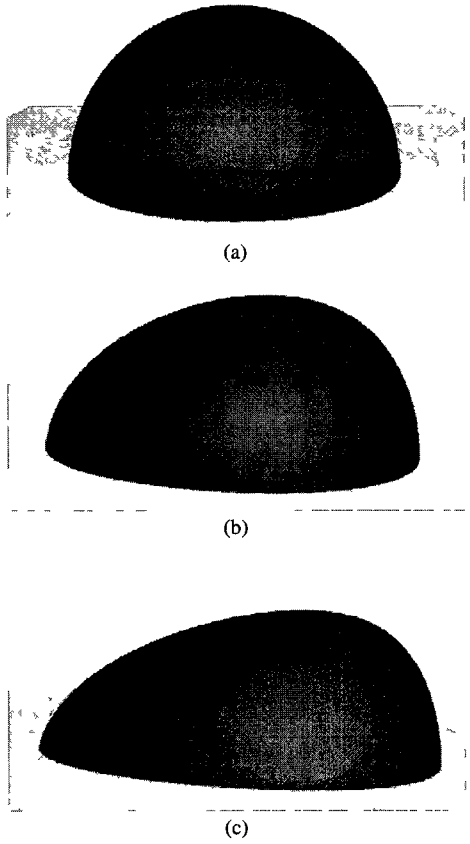


Figure 5 *Deformed drop with a) l=0 b) l=0 5 c) l=1 0*

The deformation functions are applied to each drop individually Our drop simulation method has the advantage of being divided into two parts The shape modeling and deformation are done separately from the drop dynamics (forces, velocity and position), making it easier to parallelize drop simulation We use a GPU implementation to simulate the drop motion on a surface

Unlike papers such as [Tong et al 2002] which use implicit surfaces, modeling the shape of a drop by the method presented here does not affect the shape of other drops in the neighborhood This also facilitates the modeling of individual drops in parallel The drop deformation method also has the advantage of being simpler to use than a free-form deformation model In fact only one point (the CPD) is used to deform the drop mesh

The deformation of the drop may not preserve the volume We use transformations to conserve the volume We suppose V_1 and V_2 to be the drop volume before and after the deformation The volumes V_1 and V_2 are calculated using the Equation 6 and Equation 7 By a change of variables, Equation 7 becomes Equation 8 In the Equation 8 J_h is the jacobian matrix of the function h and det is the determinant of the matrix After calculation of Equation 8 we obtain the Equation 9

$$V_1 = \int\int\int_{V_1} 1 dx_1 dy_1 dz_1 \tag{6}$$

$$V_2 = \int\int\int_{V_2} 1 dx_2 dy_2 dz_2 \tag{7}$$

$$V_2 = \int\int\int_{V_1} |det J_h(x_1, y_1, z_1)| dx_1 dy_1 dz_1 \tag{8}$$

$$V_2 \simeq V_1 + (0\ 15153) l C_1 \tag{9}$$

To conserve the volume, we simply apply a scaling to each deformed drop to satisfy the condition $V_1 = V_2$

3 3 Merging Process

The merging process is mainly based on the detection of collisions between two drops at each time step In the event of a collision one of the two drops merges with the second After this process we have only one drop with the total mass of the two initial drops

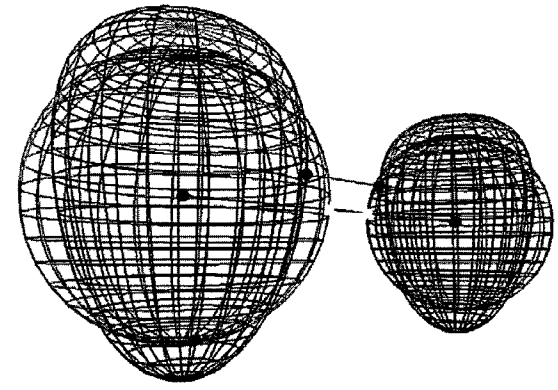


Figure 6 *Nearest distances between two drops in yellow the two nearest points before the deformation and in blue after*

The merging process uses a bounding sphere method for the collision detection Each drop has a bounding sphere The diameter of the bounding sphere is distance between the head and the tail of the drop (corresponding to O and T on Figure 3) The first step in the collision detection process is to look for drops that are in the same area on the surface An area is a triangle where is the drop and its adjacent triangles Indeed each drop is associated with a triangle on the mesh If the first step is achieved the second step is the collision detection between two bounding spheres when two drops are in the same area on the surface If the second step is achieved, the third step is to compute the smallest distance between the meshes of the

two drops This is the distance between the two nearest points on the two drops

Most collision detection algorithms accelerate calculation of the nearest distance between two object by using bounding boxes and space partitioning trees However, these tasks are not so simple to compute in real time for dynamic 3D meshes To simplify the collision detection process, we use a very effective approximation of the nearest distance between two drops The two nearest points are illustrated by the blue dots in Figure 6 To get these points, we first compute their initial position before the deformation These initial positions are given by the radius of each drop They are illustrated by the yellow dots in Figure 6 Before the deformation, the two drops are simply the red spheres shown in Figure 6 By applying the function h to two points (yellow dots), we obtain the nearest points between the two drops colored green in Figure 6, approximated by the two blue dots

Two drops collide when the nearest distance between them is below a positive value ϵ If they collide, the two drops merge to give a single drop The position of this new drop is the center of mass of the two drops in the merging process

3 4 Separation Process

In some cases a drop moving on a surface leaves behind other drops on its way The phenomenon occurs by the separation of a drop from another drop In this paper separation occurs based on the total forces acting on the drop A drop enters the separation state randomly when the norm of the total forces (denoted F) exceeds a positive value ($F \geq \mu_0$)

When a drop is in the separation state, a new drop is added on the surface This new drop has a small size compared to the initial drop The size is randomly computed The initial drop volume is reduced by the volume of the new one The position of the new drop is the same as the initial one Finally the new drop is not allowed to be in a merging state This condition prevents the merge of the new drop and the initial one instantly after the separation

3 5 Forces

Each drop is seen as a particle and its motion is governed by forces In the case of a solid object on an inclined plane, these forces are the gravitational force ($\vec{F}_g$) the surface normal reaction ($\vec{R}_N$) and the surface friction ($\vec{f}$) When the angle between the normal vector at the surface ($\vec{N}$) and $\vec{g}$ is under $\pi/2$, $\vec{R}_N$ and $\vec{f}$ are reduced to null vectors The forces $\vec{R}_N$ and $\vec{f}$ are well defined for a solid object on an inclined plane but not in the case of a fluid For a water drop, the surface tension phenomenon needs to be taken into account Surface tension is responsible for several forces [de Gennes 1985], such as the surface friction force and the internal forces between molecules that result in the drop shape Surface tension sometimes also causes a drop of water to stay glued to the surface before falling down In this paper we limit the surface tension effect by using the same surface friction force used in [de Gennes 1985] and the gluing phenomenon The other forces likes $\vec{F}_g$ and $\vec{R}_N$ are simply confined to the tangential part of $\vec{F}_g$ on the surface This tangential force is called $\vec{F}_0$ The surface friction $\vec{f}$ is simply in the opposite direction to $\vec{F}_0$ The norm f of $\vec{f}$ is given in Equation 10 The parameter σ is a standard deviation constant and v_l is the volume of the drop For the moment, the sum of the exerted forces $\vec{F}$ is simply given by $\vec{F} = \vec{F}_0 + \vec{f}$

$$f = \exp(-\frac{(1 - v_l)^4}{\sigma}) \tag{10}$$

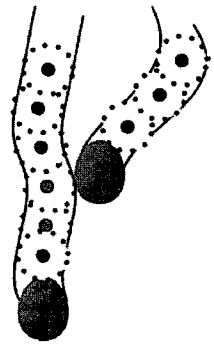


Figure 7 *Drop and trace collision illustration attractive particles are illustrated by the red points*

The motion of a drop often leaves a trail of water on the surface We call this trail of water the *trace* By observation, we note that when a drop collides with a trace it tends to continue its trajectory in the collided trace This phenomenon is simulated by using an attraction force toward the trace Some attractive particles are used in the trace of each drop in motion An attractive force denoted $\vec{F}_C$, is then used to change the trajectory of the drop to follow the collided trace To compute $\vec{F}_C$ only the attractive particles with which the drop collides in the direction of the drop motion are taken into account Let S be the set of attractive particles in red in Figure 7 Let $\vec{D}_i$ be the vector defined by $\vec{D}_i = E - P_i$ where E is the position of the drop and P_i is the position of an attractive particle The vector $\vec{F}_C$ is computed by Equation 11 The constant α is used to control how fast the drop enters the trace Figure 7 illustrates a drop and the attractive particles in red, used to compute $\vec{F}_C$ We also allow the trace to fade with time In a collision with a trace, the forces exerted become $\vec{F} = \vec{F}_0 + \vec{F}_C$ There is no surface friction when a drop is on a trace ($\vec{f} = \vec{0}$)

$$\vec{F}_C = \sum_{i \in S} \frac{\alpha}{\|\vec{D}_i\|^2} \vec{D}_i \tag{11}$$

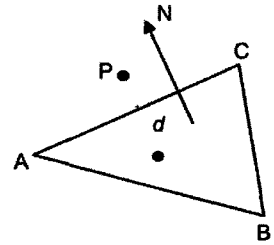


Figure 8 *Distance between a drop at the position P and the triangle ABC orthogonal projection on the triangle*

The gluing phenomenon is simulated by a force $\vec{T}$ that allows the drop of water to stay glued to the surface sometimes before falling down To simulate this phenomenon, we estimate that a drop with $\|\vec{F}\|$ less than F_T acting on it is motionless When $\|\vec{F}\|$ is more

than F_M, the drop leaves the surface Otherwise $\vec{F}$ changes by $\vec{F} \leftarrow \vec{F} + \vec{T}$

The force $\vec{T}$ tends to oblige the drop to stay on the surface It is derived from a velocity correction by an attractive constraint described in [Djado and Egli 2009] We start by computing the distance d between the drop position and the surface The distance d is a signed distance to a plane Consider a triangle ABC of the surface with $\vec{N}$ its normal vector The Figure 8 illustrates a drop at the distance d from the surface The force $\vec{T}$ is computed by Equation 12 The parameter k_T is a positive constant to control the amplitude of $\vec{T}$

$$\vec{T} = k_T d\vec{N} \tag{12}$$

The complete forces computation steps is given in Algorithm 1

Algorithm 1 *Exerted forces computation for each drop*

Compute $\vec{F_0}$ = Projection of $\vec{F_g}$ in the surface tangent plane
Compute $\vec{f}$
$\vec{F} = \vec{F_0} + \vec{f}$
if Collision with Trace **then**
  Compute $\vec{F_c}$
  $\vec{F} \leftarrow \vec{F_0} + \vec{F_c}$
end if
if $(\|\vec{F}\| < F_T)$ **then**
  $\vec{F} \leftarrow \vec{0}$ // Is Motionless on the surface
else if $(\|\vec{F}\| > F_M)$ **then**
  $\vec{F} \leftarrow \vec{F_g}$ // Is Detached from the surface
else
  Compute $\vec{T}$
  $\vec{F} \leftarrow \vec{F} + \vec{T}$ // Is in motion on the surface
end if

4 Results and Discussion

Our implementation of the drop simulation on the surface is done in the C++ language, using OpenGL for displaying the 3D The OpenGL Shading Language is also used for the simulation and the rendering of the drops on the surface To make the simulation and the initialisation faster, the 3D model of a drop is simply created using the *glutSphere* fonction from the GLUT (Graphic Language Utility Toolkit) The surface on which the drops are located is re sized in the unity box i e , each value x, y and z of each vertex of the mesh of the surface is between -1 and 1 This resizing is useful to achieve a standardization according to the parameters used for the application

The drop deformation shader is called at each frame before rendering the sphere using *glutSphere* The drop rendering shader uses an approximation of the reflection and the refraction The reflection is simply done with an environment map and the refraction by reducing the alpha channel The Fresnel equation coefficients for reflection and refraction are simulated using the angle between the normal vector and the camera direction The same shader is used for the rendering of the drop and the trace

The fluid used in the following examples is water The water density is approximately $1kg/l$ The radius of the drop before the deformation (a sphere) allows us to get the volume of the drop and its mass To simulate the drop of water, we use the parameters settings $C_1 = 1\ 0$ $C_2 = 1\ 5$ and $\lambda = 0\ 8$

The implementation of the presented method uses the CPU (Central Processing Unit) and the GPU (Graphical Processing Unit) Most parts of the drop localization on the surface (for example the index of the triangle where the drop is located) are computed on the CPU The triangle index is updated in the drop attributes In general it is simpler to get the adjacency information by a CPU approach than a GPU approach

In this paper the drop trace is modeled with a triangular mesh on the surface of the object For unification between the drop animation and the trace animation, we use a geometrical rather than a texture-based approach to render the trace With this approach the fluid is entirely represented by a 3D mesh This has the advantage of allowing us to apply other shaders or simply apply a ray tracer to render the fluid mesh (drops and traces)

Some examples are given, showing condensation on a surface and the human face sweating in real time The only external force in these examples is $\vec{g}$, but other external forces can easily be added The benchmark is done on a laptop equipped with an Intel Duo Core at 2 26GHz with an NVIDIA Geforce 9400 graphics card

4 1 Water Condensation

Water condensation is a natural phenomenon resulting from the passage of water from the gaseous to the liquid state For example, it happens when a soda can that is initially cold is exposed to ambient air This phenomenon is illustrated in Figure 1 The first two rows of Figure 9 show six frames of the water condensation simulation using our method for drop animation on a surface In this example the frame rate is around 20fps for a thousand drops The parameters used in Equation 10, Equation 11 and Equation 12 are $\sigma = 0\ 1$ $\alpha = 0\ 007$ and $k_T = 0\ 3$ The drop radius varies from $S_{Min} = 0\ 004$ to $S_{Max} = 0\ 016$ For the separation, the parame ter is set to $\mu_0 = 0\ 012$ To simulate surface tension the parameters are set to $F_M = 0\ 015$ and $F_T = 0\ 010$

4 2 Human Sweating

Human sweating is the production of particles of liquid to prevent excessive warming of the body This form of sweating first occurs in the area of the forehead before affecting the rest of the body [Eliane N and Katja 2007] This is the form of sweat which interests us here In this example, we are interested in the simulation of sweating on the human face, but this simulation model can be applied to any part of the body In Figure 1, an example of sweat drop animation is presented The last two rows of Figure 9 show six frames of the human face sweating using our drop animation method In this example the frame rate is around 17fps for a thousand drops The parameters used in Equation 10, Equation 11 and Equation 12 are $\sigma = 0\ 15$, $\alpha = 0\ 007$ and $k_T = 0\ 3$ The drop radius varies from $S_{Min} = 0\ 002$ to $S_{Max} = 0\ 007$ For the separation the parameter is set to $\mu_0 = 0\ 006$ To simulate surface tension, the parameters are set to $F_M = 0\ 007$ and $F_T = 0\ 005$

5 Conclusion

This paper present a new method for animating drops on triangular meshes in real time Unlike previous methods for real-time drop animation on surfaces we use meshes instead of normal maps to model the drops This new approach allows a simple interaction between the drops and the surface The drops may be on the surface or in the air The presented method is based on simple procedural functions implemented on the GPU to animate drops on a surface Merging and separation are used here, and simple heuristics are employed to simulate the surface tension Finally the drop trace is

simulated Collisions between drops and traces are used to simulate passage on a wet surface The method presented in this paper allows simulation of water condensation and of sweating on the human face The animation obtained is very realistic It is also simple to add levels of detail or to export the fluid mesh (drops and traces) to a more complex rendering engine

Acknowledgements

This research was supported in part by grants from the Natural Sciences and Engineering Research Council of Canada to the second author The authors would like to thank Gilles-Philippe Paille, Fabrice Colin, Martin Guay, Christian Preciado Castelo and Olivier Vaillancourt for their comments

References

ALGAN, E , KABAK M , OZGUC, B AND CAPIN, T 2008 Simulation of water drops on a surface In *3DTV Conference The True Vision Capture Transmission and Display of 3D Video*, 361–364

DE GENNES P G 1985 Wetting statics and dynamics *Rev Mod Phys 57* 3 (Jul), 827–863

DJADO K , AND EGLI R 2009 Particle-based fluid flow visualization on meshes In *AFRIGRAPH 09 Proceedings of the 6th International Conference on Computer Graphics Virtual Reality Visualisation and Interaction in Africa* ACM, New York, NY, USA, 65–72

DORSEY J PEDERSEN, H K AND HANRAHAN P 1996 Flow and changes in appearance In *SIGGRAPH 96 Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* ACM, New York, NY, USA 411–420

EL HAJJAR, J -F , JOLIVET, V , GHAZANFARPOUR D AND PUEYO X 2009 A model for real-time on-surface flows *Vis Comput 25*, 2, 87–100

EL HAJJAR, J -F 2008 Simulation decoulements liquides sur des surfaces solides pour lanimation en synthese dimages Phd thesis, FACULTE DES SCIENCES ET TECHNIQUES, ECOLE DOCTORALE (Science Technologie Sante) UNIVERSITE DE LIMOGES October

ELCOTT S TONG Y , KANSO, E , SCHRODER, P , AND DESBRUN, M 2007 Stable circulation-preserving, simplicial fluids *ACM Trans Graph 26*, 1, 4

ELIANE N M AND KATJA H 2007 In *Human Anatomy and Physiology*

FAN, Z ZHAO Y KAUFMAN, A AND HE Y 2005 Adapted unstructured lbm for flow simulation on curved surfaces In *SCA 05 Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM Press, New York, NY, USA 245–254

FOURNIER P , HABIBI, A AND POULIN, P 1998 Simulating the flow of liquid droplets In *Proceedings of the Graphics Interface Conference*, Canadian Human-Computer Communications Society, 133–142

IGLESIAS, A , GALVEZ, A , AND PUIG-PEY, J 2001 Generating drop trajectories on parametric surfaces *J Y Q Peng W Li (ed) Proceedings of the Seventh CAD/Graphics 2001*, 350–357

JUNG, Y , AND BEHR, J 2009 Gpu-based real-time on-surface droplet flow in x3d In *Web3D 09 Proceedings of the 14th International Conference on 3D Web Technology*, ACM, New York NY, USA, 51–54

KANEDA K , KAGAWA T , AND YAMASHITA, H 1993 Animation of water droplets on a glass plate *Proc Computer Animation*, 177–189

KANEDA, K , ZUYAMA, Y , YAMASHITA, H , AND NISHITA, T 1996 Animation of water droplet flow on curved surfaces *Proc PACIFIC GRAPHICS 96*, 50–65

KANEDA, K , IKEDA, S AND YAMASHITA, H 1999 Animation of water droplets moving down a surface *Journal of Visualization and Computer Animation 10* 1 15–26

LUI, L , WANG, Y AND CHAN T F 2005 Solving pdes on manifold using global conformal parameterization In *Variational Geometric and Level Set Methods in Computer Vision Third International Workshop VLSM 2005* 307–319

MURTA A AND MILLER, J 1999 Modelling and rendering liquids in motion In *WSCG99 Conference Proceedings* V Skala (ed), 194–201

SATO, T DOBASHI Y AND YAMAMOTO T 2002 A method for real-time rendering of water droplets taking into account interactive depth of field effects In *IWEC* 125–132

SHI, L , AND YU, Y 2004 Inviscid and incompressible fluid simulation on triangle meshes *Comput Animat Virtual Worlds 15* 3-4, 173–181

STAM, J 2003 Flows on surfaces of arbitrary topology In *SIGGRAPH 03 ACM SIGGRAPH 2003 Papers* ACM New York NY, USA, 724–731

TAKENAKA S MIZUKAMI Y AND TADAMURA K 2008 A fast rendering method for water droplets on glass surfaces In *The 23rd International Technical Conference on Circuits/Systems Computers and Communications (ITC CSCC)*, 13–16

TONG, R , KANEDA, K AND YAMASHITA, H 2002 A volume-preserving approach for modeling and animating water flows generated by metaballs *The Visual Computer 18*, 8, 469–480

WANG, H , MUCHA, P J , AND TURK G 2005 Water drops on surfaces In *SIGGRAPH 05 ACM SIGGRAPH 2005 Papers*, ACM New York, NY, USA, 921–929

YANG Y , ZHU, C AND ZHANG, H 2004 Real-time simulation Water droplets on glass windows *Computing in Science and Engg 6* 4, 69–73

YU, Y , JUNG, H , AND CHO, H 1998 A new rendering technique for water droplet using metaball in the gravitation force In *WSCG98 Conference Proceedings*, V Skala (ed) 432–439

YU, Y -J , JUNG, H -Y , AND CHO, H -G 1999 A new water droplet model using metaball in the gravitational field *Computers & Graphics 23*, 2, 213–222

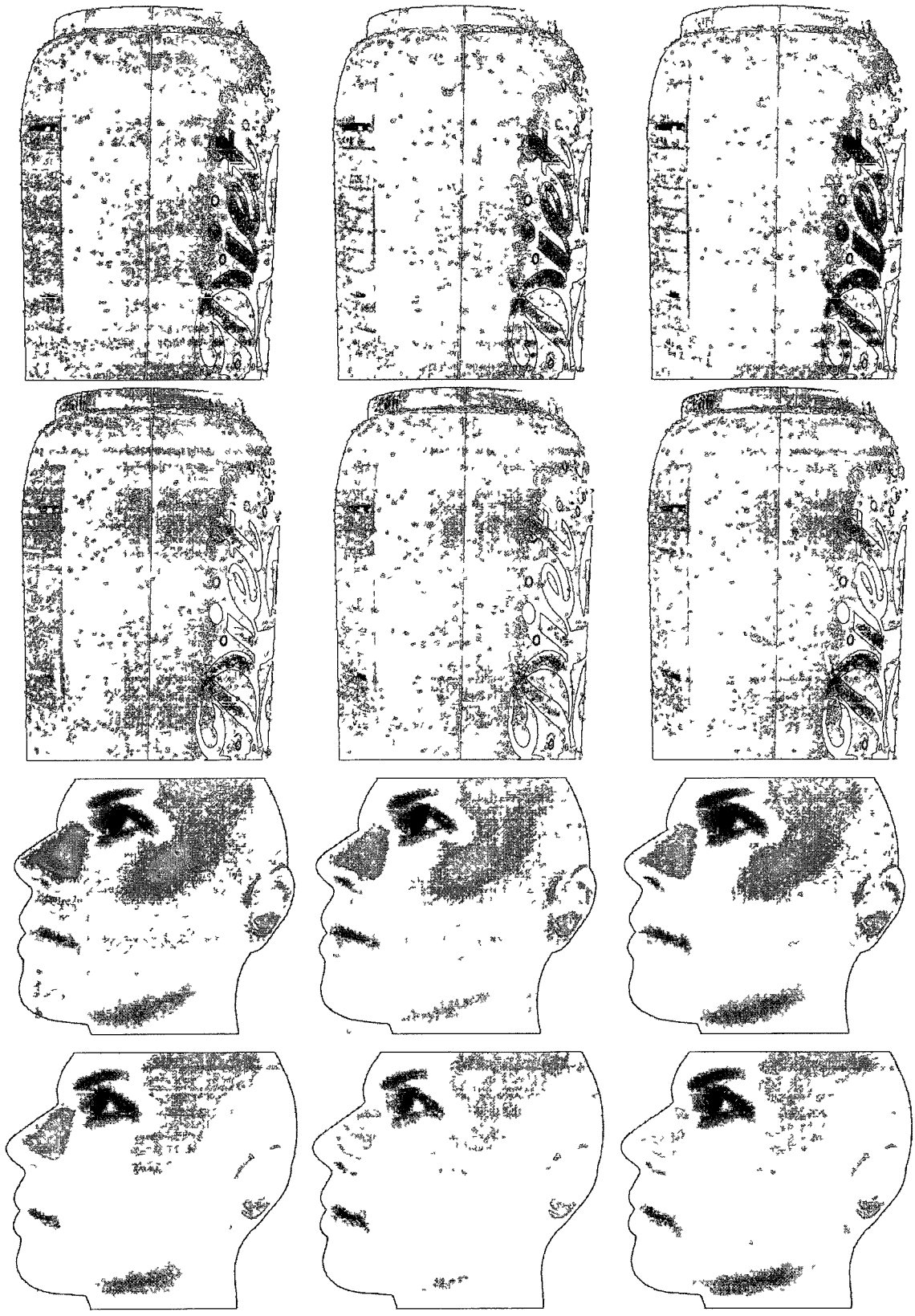


Figure 9 *Images from a water condensation application on the two first rows Images from a human face sweating application on the two last rows*

Conclusion

Dans cette thèse, nous avons introduit de nouvelles méthodes d'animation de fluide Ce domaine peut être simplement un volume ou bien la surface d'un objet 3D Nos contributions et limitations spécifiques à chaque chapitre peuvent être résumées comme suit

Dans le premier chapitre, nous avons présenté une nouvelle méthode de simulation d'un fluide par interpolation de vitesses précalculées De bons résultats ont été obtenus en comparaison avec la méthode par différences finies, tout en permettant des meilleures performances Nous avons pu visualiser le champ de vitesses obtenu avec plusieurs méthodes, y compris avec des particules La méthode présentée dans le chapitre 1 peut être appliquée dans un cadre de simulation de fluide en temps réel comme les jeux vidéo La méthode présentée est très facile à implémenter et est parallélisable Comme principale limitation, la méthode présentée n'est valide que pour une seule source de vitesse appelée «blowers» Il serait donc intéressant de pouvoir simuler la présence de plusieurs blowers

Dans le second chapitre, nous avons présenté une nouvelle méthode de visualisation d'un fluide sur la surface d'un objet avec des particules Ces particules ont été utilisées pour faire le rendu de liquide ou bien de la fumée sur la surface Les résultats obtenus sont originaux et représentent une nouveauté dans la simulation de fluide sur une surface Notre méthode de visualisation de fluide sur une surface présente toutefois des limites En effet, la distance entre deux particules utilisée pour le calcul de la répulsion est une approximation Il serait intéressant d'utiliser les distances exactes entre les particules sur la surface par une approche géodésique L'utilisation des distances géodésiques est complexe et prend beaucoup plus de temps, mais cette approche permet de réduire la dépendance de la distribution des particules et la résolution du maillage de la surface Il serait intéressant d'adapter la méthode d'interpolation utilisée pour la simulation du fluide du chapitre 1 sur la surface pour une simulation avec un champ de vitesses dynamique

Dans le dernier chapitre, nous avons présenté une nouvelle méthode de simulation en temps réel de gouttes d'eau sur une surface La représentation géométrique des gouttes leur permet d'être libres, de rester sur la surface ou bien dans l'air Les résultats obtenus sont très réalistes Nous avons pu traiter des phénomènes physiques comme la condensation de l'eau sur des surfaces ou encore la sueur sur un visage Comme principale amélioration, nous pensons qu'il serait intéressant de pouvoir simuler sur une surface déformable tout en améliorant la performance de l'application En effet, une goutte sur une surface qui est en mouvement subit l'effet d'autres forces extérieures dont il faut tenir compte Il serait également intéressant de pouvoir paralléliser les étapes de recherche de voisinage entre triangles, notamment avec le processeur graphique Dans notre implémentation, nous n'avons pas été capables d'implémenter tous nos algorithmes sur le processeur graphique

Les méthodes développées dans cette thèse représentent des éléments importants pour l'animation de fluide en infographie sur une surface Les résultats obtenus dans ces méthodes ont permis de créer des effets spéciaux de fluide sur une surface Une prochaine étape serait de bâtir un simulateur d'orage en temps réel Le lieu de l'orage serait un champ de vitesses 3D provenant du chapitre 1 Les chapitres 2 et 3 permettront de considérer la pluie comme des gouttes d'eau qui vont interagir avec les surfaces des objets dans la scène

Bibliographie

[1] E ALGAN, M KABAK, B OZGUC et T CAPIN « Simulation of Water Drops on a Surface » Dans 3DTV Conference The True Vision - Capture, Transmission and Display of 3D Video, pages 361–364, 2008

[2] D BAUER, R PEIKERT, M SATO et M SICK « A case study in selective visualization of unsteady 3D flow » Dans VIS '02 Proceedings of the conference on Visualization '02, pages 525–528, Washington, DC, USA, 2002 IEEE Computer Society

[3] R BRIDSON Fluid Simulation for Computer Graphics A K Peters, 2008

[4] R EGLI et N F STEWART « Chain models in computer simulation » Math Comput Simul , 66(6) 449–468, 2004

[5] J -F EL HAJJAR, V JOLIVET, D GHAZANFARPOUR et X PUEYO « A model for real-time on-surface flows » Vis Comput , 25(2) 87–100, 2009

[6] S ELCOTT, Y TONG, E KANSO, P SCHRODER et M DESBRUN « Stable, circulation-preserving, simplicial fluids » ACM Trans Graph , 26(1) 4, 2007

[7] Z FAN, Y ZHAO, A KAUFMAN et Y HE « Adapted unstructured LBM for flow simulation on curved surfaces » Dans SCA '05 Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation, pages 245–254, New York, NY, USA, 2005 ACM Press

[8] N FOSTER et D METAXAS « Realistic animation of liquids » Graph Models Image Process , 58(5) 471–483, 1996

[9] K KANEDA, T KAGAWA et H YAMASHITA « Animation of water droplets on a glass plate » Dans Proc Computer Animation, pages 177–189, 1993

[10] J KRUGER, P KIPFER, P KONDRATIEVA et R WESTERMANN « A Particle System for Interactive Visualization of 3D Flows » IEEE Transactions on Visualization and Computer Graphics, 11(6) 744–756, 2005

[11] L M LUI, Y WANG et Tony F CHAN « Solving PDEs on Manifold using Global Conformal Parameterization » Dans Variational, Geometric, and Level Set Methods in Computer Vision Third International Workshop, VLSM 2005, pages 307–319, Beijing, China, 2005

[12] N MAX, R CRAWFIS et C GRANT « Visualizing 3D velocity fields near contour surfaces » Dans VIS '94 Proceedings of the conference on Visualization '94, pages 248–255, Los Alamitos, CA, USA, 1994 IEEE Computer Society Press

[13] G -P PAILLÉ Simulation d'un fluide à la surface d'un objet par la méthode 'smoothed particle hydrodynamics' M Sc Thèses de l'Université de Sherbrooke - Sciences - Informatique, 2009

[14] L SHI et Y YU « Inviscid and incompressible fluid simulation on triangle meshes » Comput Animat Virtual Worlds, 15(3-4) 173–181, 2004

[15] J STAM « Flows on surfaces of arbitrary topology » Dans SIGGRAPH '03 ACM SIGGRAPH 2003 Papers, pages 724–731, New York, NY, USA, 2003 ACM

[16] S TAKENAKA, Y MIZUKAMI et K TADAMURA « A Fast Rendering Method for Water Droplets on Glass Surfaces » Dans The 23rd International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC), pages 13–16, 2008

[17] J J van WIJK « Image based flow visualization » Dans SIGGRAPH '02 ACM SIGGRAPH 2002 Papers, pages 745–754, New York, NY, USA, 2002 ACM Press

[18] H WANG, P J MUCHA et G TURK « Water drops on surfaces » Dans SIGGRAPH '05 ACM SIGGRAPH 2005 Papers, pages 921–929, New York, NY, USA, 2005 ACM

[19] Y YANG, C ZHU et H ZHANG « Real-Time Simulation Water Droplets on Glass Windows » Computing in Science and Engg , 6(4) 69–73, 2004

[20] Y -J YU, H -Y JUNG et H -G CHO « A new water droplet model using metaball in the gravitational field » Computers & Graphics, 23(2) 213–222, 1999