EG UK Theory and Practice of Computer Graphics (2007)
Ik Soo Lim, David Duce (Editors)

Real-time Fluid Simulation Coupled with Cloth

Takahiro Harada, Seiichi Koshizuka and Yoichiro Kawaguchi

The University of Tokyo

Abstract

This paper presents a real-time simulation method for coupling of cloth and fluids computed by using Smoothed Particle Hydrodynamics (SPH). To compute interaction between cloth consisting of several polygons and fluid particles, the distance between cloth to the particle have to be calculated. It is computationally expensive because we have to compute the distance to the faces, edges and vertices of polygons. Instead of calculating the exact distance to a cloth, we calculate an approximate distance by using the distance to the faces and the gravitational centers of the polygons. This paper also presents techniques to perform the coupled simulation entirely on Graphics Processing Units (GPUs). The computation of interaction forces is divided into fluid-cloth and cloth-fluid forces to implement entire simulation on GPUs. By exploiting the parallelizm of GPUs, we could couple simulations of several tens of thousands of fluid particles and cloth which consists of several thousands of polygons in real-time.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Physically based Modeling; I.3.7 [Computer Graphics]: Animation

1. Introduction

As the motions of fluids are complex, it is difficult to make animations of these manually. The interaction between fluids and rigid and elastic bodies increases the complexity of the motion of fluids. However, we can compute their motion using physically based simulations. Faster simulations are needed for real-time applications such as games. Although physically based simulations have been well studied in the field of computational mechanics, methods developed in this field are not applicable to real-time applications without modification. This is because they place emphasis on accuracy but not on speed. We have to modify the algorithms or increase the efficiency of computation to apply them to real-time applications. Since modern processors are shifting toward parallel architecture, how to apply simulation methods to these platforms and exploit their computational power is an important research theme for real-time applications.

There are two methods that simulate free surface flow. One is the Eulerian method, which uses mesh and the other is the Lagrangian method, which uses particles. Because the advection term is calculated by advecting particles themselves which represent a bunch of fluids, the Lagrangian method does not suffer from numerical dissipation caused by advection. This is an important point for computer graphics, especially for real-time applications, because numerical dissipation causes mass dissipation of fluids. Therefore, the Lagrangian methods are well suited to real-time applications.

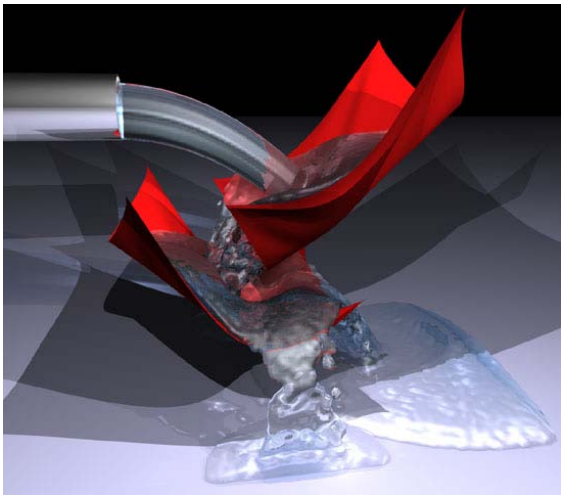

Figure 1: *Real-time simulation of cloth and fluid interaction. A fluid is poured onto a sheet of cloth. This simulation runs about 14 frames per second on GeForce 8800GTX.*

In this paper, we present a real-time simulation method for the coupling of fluids and cloth. Smoothed Particle Hydrodynamics (SPH), which is one of the Lagrangian methods, is used for fluid simulation. The distance between them have to be computed to compute the interaction between cloth and fluids. However, the computation is intricated and computationally expensive, we calculate an approximate distance by using the distance to faces and the gravitational centers of the polygons. This paper also presents techniques to implement the simulation entirely on Graphics Processing Units (GPUs). The interaction computation which cannot be implemented straight-forward on GPUs, is performed on GPUs by dividing it into two operations, the computation of the force from cloth to fluid and from fluid to cloth. By exploiting the parallelizm of GPUs, the speed of simulation is accelerated dramatically. Scenes such as shown in Figure 1 can be simulated in real-time.

2. Related Works

Since Foster and Metaxas first introduced three-dimensional fluid simulation to solve the Navie-Stokes equation to the computer graphics community [FM96], there have been many studies that have applied the Eulerian method to computer graphics [Sta99]. There have also been several studies of free surface flow that use the Level set method [FF01] [EMF02] [LSSF06]. Klinger *et al.* used tetrahedral meshes and remeshed the computational domain dynamically [KFCO06]. Since these simulation methods for free surface flow are computationally expensive, they are not applicable to real-time applications. On the other hand, there have also been several studies of Lagrangian methods. Müller *et al.* used SPH for real-time applications [MCG03] and the used several thousands of particles in real-time. Müller *et al.* also applied SPH to multi-phase flow [MSKG05]. Kipfer *et al.* studied a river simulation by using a data structure suited for sparse particle distribution [KW06].

A simulation of coupling of fluids and other objects is a much intricate topic. Génevaux *et al.* studied coupling of fluids and elastic bodies represented by particles and springs [GHD03]. These simulations were coupled by computing the force between marker particles of fluids and particles of elastic bodies. Müller *et al.* simulated the coupling of SPH fluid and elastic bodies represented by tetrahedral meshes [MST*04]. The interaction between them was computed by generating temporary particles on the surface of tetrahedral meshes. They simulated a few thousand of fluid particles in real-time. Guendelman *et al.* coupled fluid simulation and thin shells [GSLF05]. However, as their method is computationally expensive, it is difficult to apply on real-time applications. Chentanez *et al.* extended the method developed by Klingner *et al.* and developed a strong coupling method for fluids and elastic bodies.

As the growth of the computational power of GPUs has been tremendous, many researchers have tried to use them for general purpose computations [OLG*05]. There are image space collision detection techniques [VSC01] [GRLM03]. However, their accuracy is governed by the resolutions of images. Harris *et al.* accelerated a Coupled Map Lattice and Wei *et al.* and Li *et al.* studied Lattice-Boltzmann Method for real-time fluid simulation [HCSL02] [WLMK04] [LFWK03]. There are also studies which used GPUs to accelerate Eulerian fluid simulation [HBSL03] [LLW04]. Although there is no study which accelerate free surface flow by Eulerian methods, some researchers used GPUs to accelerate Lagrangian free surface simulation. Amada *et al.* accelerated SPH [AIY*04], but they could not exploit the computational power of GPUs because neighboring particles were searched on CPUs. A method presented by Kolb *et al.* accompanied numerical dissipation because the physical values on particles were computed by interpolation of grid values [KC05]. These issues are overcomed by a study by Harada *et al.* and they demonstrated that SPH can be accelerated drastically on GPUs [HKK07].

3. Fluid Simulation

The governing equations for incompressible flow are the mass and momentum conservation equations.

$$\frac{D\rho}{Dt} = 0 \tag{1}$$

$$\frac{D\mathbf{U}}{Dt} = -\frac{1}{\rho}\nabla P + \nu\nabla^2\mathbf{U} + \frac{\mathbf{F}}{\rho}, \tag{2}$$

where $\rho, \mathbf{U}, P, \nu, \mathbf{F}$ are the density, pressure, velocity, kinematic viscosity coefficient of fluid and external force, respectively. In this study, fluids are discretized to a set of particles and the governing equations are solved using SPH. Although the SPH does not solve Equation (1) and so it can not solve incomplessible flow, it can compute near incompressible flow by lowering the compressibility. A physical value at a point is calculated by a weighted sum of particle values. A computational model used by Müller *et al.* is employed in this study [MCG03].

4. Cloth Simulation

Since the interacting force from fluids to cloth is also calculated as an external force, we can use any simulation models for the cloth. Here, we employ a mass-spring model in which particles are connected by two kinds of springs as shown in Figure 2. For time integration, the verlet method is employed. This method is suited to real-time simulation, because it is computationally inexpensive and has better numerical stability than the explicit Euler method. The position $\mathbf{x}_i^{t+dt}$ of particle i at $t+dt$ is computed as follows:

$$\mathbf{x}_i^{t+dt} = \mathbf{x}_i^t + d(\mathbf{x}_i^t - \mathbf{x}_i^{t-dt}) + \frac{\mathbf{F}_i dt^2}{m_i}, \tag{3}$$

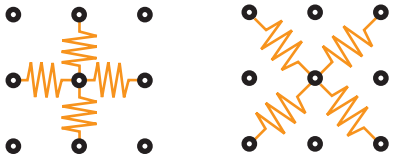

Figure 2: *Two kinds of springs of cloth model.*

where m_i, d, dt are the mass of particle i, damping coefficient and time step, respectively. An external force $\mathbf{F}_i$ consists of gravity $m_i\mathbf{g}$, spring force $\mathbf{F}_{i,spring}$ and the force from the fluid $\mathbf{F}_{i,fluid}$. The spring force $\mathbf{F}_{i,spring}$ from two kinds of springs is calculated as follows

$$\mathbf{F}_{i,spring} = \sum_{j \in N_{adj}} k_{adj}(|\mathbf{x}_{ij}| - l_{adj}) \frac{\mathbf{x}_{ij}}{|\mathbf{x}_{ij}|} + \sum_{j \in N_{diag}} k_{diag}(|\mathbf{x}_{ij}| - l_{diag}) \frac{\mathbf{x}_{ij}}{|\mathbf{x}_{ij}|}, \quad (4)$$

where k_{adj}, l_{adj} are the spring coefficient and the rest length of springs connecting adjacent particles, respectively, and k_{diag}, l_{diag} are those of springs connecting diagonal particles. $\mathbf{x}_{ij}$ is the relative position of particle j from particle i.

5. Coupling of Fluid and Cloth

5.1. Spatial Division

To calculate the force on a particle, we have to search for neighboring particles. The computational cost is $O(n^2)$ if they are searched for from all of the particles. The efficiency of the computation is improved by introducing a three-dimensional grid covering the computational domain [Mis03]. A particle index is stored in the voxel to which the particle belongs. With the grid, neighboring particles of particle i are restricted to particles whose indices are stored in the voxel in which index i is stored or voxels adjacent to the voxel.

When the interactions between fluid particles and cloth polygons are calculated, the polygon that is the closest to a particle have to be searched for as well. Because the computational cost of the brute-force method is proportional to the number of particles multiplied by the number of polygons, a uniform grid is also introduced to reduce the computational cost. Each voxel stores the indices of polygons whose gravitational center is inside of the voxel. Thus, two grids, one is for fluid particles and the other for cloth polygons, are used for the simulation as shown in Figure 3.

In this study, all the polygons belonging to a cloth are of the same size at the rest state and they do not change their sizes very much during the simulation because of the physical properties of the cloth. Therefore, the side lengths of voxels are adjusted by the rest distance between the gravitational centers of polygons.

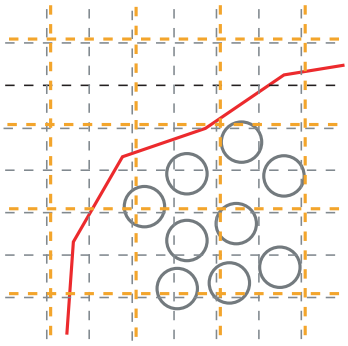

Figure 3: *Two grids. Gray-dashed lines indicate a grid for fluid particles and orange-dashed lines indicate a grid for cloth.*

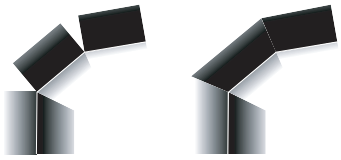

Figure 4: *Discontinuous force field (left) and continuous force field (right).*

5.2. Collision Detection

The interacting force between fluid particles and cloth is calculated from the distance from each particle to the cloth. Firstly, the closest polygon is searched for from all of the polygons because the distance to the cloth is the distance to the closest polygon belonging to the cloth. Although this operation is computationally expensive, we do not have to search for from all of them since the grid has already been generated. Polygons closer to a particle are restricted to polygons whose indices are stored in voxels adjacent to the voxel to which the particle belongs. Therefore, we calculate the distance from the particle to voxels whose indices are stored in the 27 voxels.

Assuming that the cloth has a certain thickness, the interaction force between a fluid particle and cloth is modeled as a force proportional to the penetration depth. However, the force field in the computational domain becomes discontinuous at the connections of polygons as shown in the left side of Figure 4 when only the distance from the face of a polygon is computed. This discontinuity results in an artifact of fluid particle motion. Although we can eliminate this problem by computing the exact distance from the cloth, i.e., the distance from edges and vertices belonging to polygons, this also increases the computational cost. Therefore, we compute an approximate distance to the cloth.

Assuming that there are n polygons $T = \{t_0, t_1, \cdots t_n\}$ near a particle. We first compute the distance d_c to the gravita-

tional center of polygon and a polygon which has the smallest value d_c is selected to the closest polygon $t_{closest}$. Let the position of three vertices of polygon be $j, \mathbf{v}_{j,0}, \mathbf{v}_{j,1}, \mathbf{v}_{j,2}$. The condition is written as follows.

$$t_{closest} = arg \min_{t_j \in T} (\mathbf{x} - \frac{\mathbf{v}_{j,0} + \mathbf{v}_{j,1} + \mathbf{v}_{j,2}}{3})^2 \quad (5)$$

We assumed that these three vertices have the same mass and so that the position of the gravitational center can be calculated as the average position of the three vertices.

In the next step, the distance d_p to the polygon $t_{closest}$ is calculated with the vertex positions as follows.

$$d_p = |(\mathbf{v}_{j,1} - \mathbf{v}_{j,0}) \times (\mathbf{v}_{j,2} - \mathbf{v}_{j,0}) \cdot (\mathbf{x} - \mathbf{v}_{j,0})| \quad (6)$$

Because d_p is continuous in the computational domain, the calculated force field is also continuous as shown in the right of Figure 4. A particle is colliding to a cloth when the distance to a cloth is smaller than a thickness of the cloth ε. In this study, ε is set to the rest fluid particle distance. There is also discontinuity of the force field at the edge of cloth. However, this discontinuity does not produce severe artifacts, they are not considered further.

5.3. Fluid Reaction

The pressure term of the fluid works as a force that makes the density of the fluid constant, i.e., keeps the distance between particles the rest distance. The force from the cloth is also modeled in the same manner. When the distance d to the cloth is smaller than the rest distance ε, a force is introduced to bring the particle back to the distance ε. The force $\mathbf{f}_i^{press}$ is calculated with the normal vector $\mathbf{n}$ of the polygon as follows:

$$\mathbf{f}_i^{press} = m_i \frac{(\varepsilon - d)\mathbf{n}}{dt^2}. \quad (7)$$

When a cloth is within the effective radius from a particle, there must be a viscosity force. Assume that particles are placed on the surface of a cloth perpendicular to the vector to the cloth and they have uniform velocities $\mathbf{v}$ and densities ρ. The viscosity force from the cloth is modeled as follows:

$$\mathbf{f}_{i,wall}^{vis} = -\mu \frac{m}{\rho} (\mathbf{v} - \mathbf{v}_i) \sum_{j \in Wall} \nabla W_{vis}(\mathbf{r}_{ij}). \quad (8)$$

Assuming that the particles on the cloth are placed uniformly, their distribution is specified when the distance to the cloth is decided. Therefore, the sum of the weight function can be precomputed and the value is read at the simulation.

The contribution to the density has to be estimated, as well as the viscosity term. Since there are no particles on the cloth, the density of a fluid particle near the cloth is given a lower value. This leads to concentration of particles near the cloth. To overcome this problem, the contribution to the density of the cloth is calculated as follows. With the same assumption we had in the viscosity term, the contribution of density $\rho_{i,wall}$ as a weighted sum of these particles is

$$\rho_{i,wall} = m \sum_{j \in Wall} W(\mathbf{r}_{ij}). \quad (9)$$

These values are also precomputed because their distribution is decided according to the distance to the cloth.

5.4. Cloth Reaction

The fluid force on a polygon belonging to a cloth works in the direction of the pressure gradient of a fluid. The translational and rotational motions of a polygon are derived from the force. However, the force does not induce a rotational motion when the direction is perpendicular to the polygon. In that case, the fluid force can be calculated as a force that works at the gravitational center of the polygon. When a cloth is consist of a large number of sufficiently small polygons, the variation of the pressure of a fluid over the polygon is negligible and so the direction of the pressure gradient can be approximated by the direction of the normal vector of the polygon. Thus, the rotational motion induced by the fluid is also negligible. In this study, the cloth reaction is modeled assuming that polygons belonging to a cloth is small enough to use this assumption.

From the Newton's third law, the force $\mathbf{F}_i$ on polygon i is the negative sum of the colliding forces $\mathbf{f}_j$ on the fluid particles. The force $\mathbf{F}_i$ on polygon i is calculated as

$$\mathbf{F}_i = - \sum_{j \in Colliding} \mathbf{f}_j. \quad (10)$$

By the assumption described above, the force from fluid does not induce a rotational motion of a polygon. Thus, the force $\mathbf{F}_i$ is distributed uniformly among the vertices $\mathbf{f}_{i,0}, \mathbf{f}_{i,1}, \mathbf{f}_{i,2}$ belonging to the polygon i.

$$\mathbf{f}_{i,0} = \mathbf{f}_{i,1} = \mathbf{f}_{i,2} = \frac{\mathbf{F}_i}{3} \quad (11)$$

6. Acceleration using GPUs

6.1. Data Structure

Physical values of fluids and cloth have to be stored in two-dimensional floating point textures. Since the values that a fluid particle possess include position, velocity and density, textures are prepared for each of them and one texel represents a particle. Cloth has position and connectivity information for particles. Textures are also prepared for them. The vertex positions and connectivity is stored as shown in Figure 5. Other than these physical values, we have to allocate memories for the two grids. Although the latest GPU supports writing to a three-dimensional texture, there are performance advantages in using two-dimensional texture. Therefore, a flat 3D texture, which is a technique to store a three-dimensional texture in a two-dimensional texture, is used to store the three-dimensional grid [HBSL03].

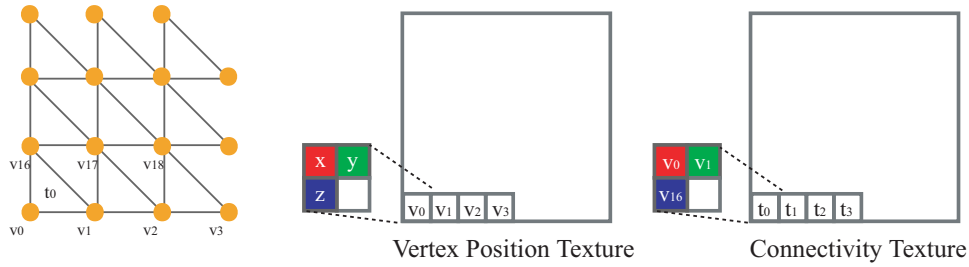

Figure 5: *Data structures of the vertrex positions and connectivity of a cloth. The vertex position texture store positions of vertices and the connectivity texture stores vertex indices which a polygon consists of.*

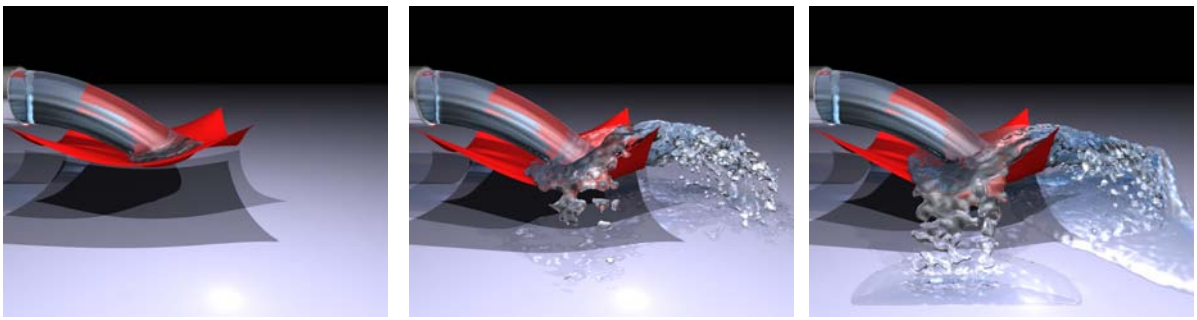

Figure 6: *Results of a simulation in which a fluid is poured onto two sheets of cloth.*

6.2. Grid Generation

To search for neighboring particles and polygons, grids have to be generated, which is done by preparing vertices for each particles and moving the vertices to the grid coordinates of the corresponding particles. This procedure cannot store indices into the grid correctly when several indices of particles have to be stored in a voxel. Therefore, we used a method proposed by [HKK07], with which we can store indices correctly when there are several particles. The grid for a cloth is constructed in the same way to that for fluid particles. The only difference is that a vertex is assigned to each polygon belonging to a cloth.

6.3. Fluid Simulation

To compute fluid motion, the density has to be calculated first and then the pressure force is computed with the calculated densities. The velocity of the fluid is updated with the pressure, viscosity and external forces by Equation (2). Finally, the position is updated. Since a texture coordinate is assigned to each fluid particle and the physical values are stored at the coordinate in textures, most of the operation is done from reading a value at the texture coordinate in textures and writing a value to another texture except for the weighted sum of values of neighboring particles. To compute this, indices of neighboring particles are read from the grid texture. Then the values of these particles are read from the textures and the weighted sum is evaluated.

6.4. Interaction Force Computation

We first describe the calculation of the pressure force. The pressure force is computed by two operations because the forces on fluids and cloth cannot be calculated at the same time when data-parallel processors, such as GPUs, are used. Therefore, in the first step, the closest polygon to a particle is searched for with the grid storing the indices of cloth polygons. A coordinate of a particle in the cloth grid is calculated and the indices of polygons stored around the voxel to which the particle belongs are read. Then, the distances to these polygons are calculated as described in Section 5.2. The distance to the closest polygon and the index of the polygon are written into a texture. If the distance to the cloth is smaller than ε, the pressure force on the particle is computed with Equation (7).

The force on the polygon is then calculated. This force is computed by searching for the neighboring particles as described above. However, we have to calculate the distance to a larger number of particles because the sizes of polygons are larger than fluid particles in the most cases. Our simulation is not an exception, therefore we took an alternative strategy. We already know the polygon that collides with the fluid particles. Thus, the reaction force on a polygon can be

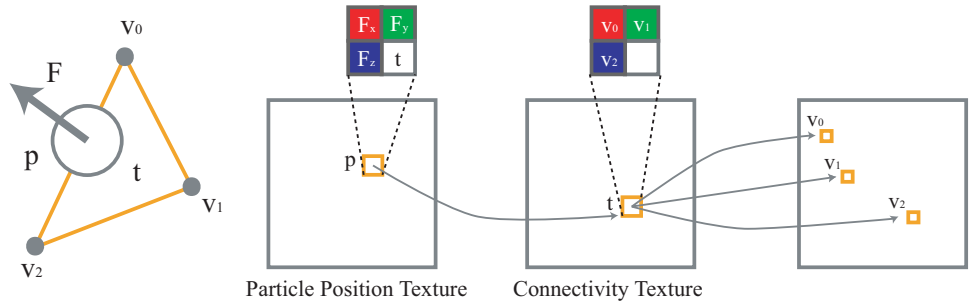

Figure 7: *Computation of the force on a polygon. When particle p is colliding with polygon t consisting of v_0, v_1, v_2, force* **F** *is stored in a particle force texture. At the same time, polygon index t is also written in the texel. From the polygon index t, the indices of vertices v_0, v_1, v_2 are read from the connectivity texture. Then the force* **F** *is distributed among these vertices whose texture coordinates are calculated with the indices v_0, v_1, v_2.*

Table 1: *Number of fluid particles and frame rates. Results of fluid simulation without coupling are shown in the bottom row.*

Number of particles	16,384	65,536
Figure 1	45.7	12.3
Figure 6	53.3	14.9
Figure 8	49.2	13.1
Fluids	64.1	16.6

computed without searching for colliding particles. Instead, a force acting on a particle is summed up at each polygon and we obtain the forces on each of them. Because of the data structure employed, in which a force on cloth is stored as forces on the vertices belonging to it, a colliding force on a particle is distributed among these vertices. Therefore, this operation is performed by rendering vertices, prepared for each particle, to the position of vertices belonging to colliding polygons as one pixel sprite and output the forces read from the particle force texture as colors as shown in Figure 7. Because the vertex shader cannot write the force to three pixels simultaneously, three vertices are prepared for each particle and the forces to three vertices are written sequentially.

As described in Section 5.3, the weighted sums of the cloth contribution for the viscosity term and density can be computed in advance. The values are calculated at some points in the effective radius and stored in a texture. The value corresponding to the distance is read from it in the simulation. At other points, the values are calculated by linear interpolation.

7. Results

The present method was implemented on a PC with a Core 2 X6800 CPU and a GeForce 8800GTX GPU. The programs were written in C++, OpenGL and C for Graphics. All of the results shown in this section were rendered after simulations. Blobs are used for fluid particles and the surface is extracted using Marching Cubes [LC87].

Figure 6 shows the results of a simulation where a fluid is poured onto a cloth. Balls of fluid are dropped onto a cloth in Figure 8. We can see that the fluid changes the shape of cloth. Figure 1 shows the results of a simulation with two sheets of cloth. In these simulations, 65,536 fluid particles were used and a cloth consisted of 8,192 polygons (the total number of polygons is 16,384). The frame rates were approximately 14 frames per second as shown in Table 1. These frame rates are measured with a rendering using point sprites for particles. When 16,384 fluid particles were used in the same scene as Figures 6 and 8, the frame rates were about 50 frames per second. The frame rates of fluid simulation without coupling are also shown in the bottom of Table 1. Since the difference in frame rates between a fluid simulation alone and a coupled simulation is small, the computational cost of coupling is smaller than the fluid simulation itself.

Our method has a limitation in that, as the method does not compute collisions between the cloths, they cannot interact with each other. Self collision and cloth-cloth collision computations will be conducted as future works. The proposed collision computation method which uses a grid assumed that the size of polygons are almost uniform. If there is a large deformation of a polygon or variation of the size, we have to increase the number of voxels which colliding particles are searched for. The force on a cloth is computed by assuming the size of a polygon is small enough to approximate that there is no variation of the pressure of a fluid over the polygon. As the size of a polygon get larger and larger, the error by this assumption increases. Therefore, our method to compute the force on a cloth is not applicable to a cloth with a larger polygon.

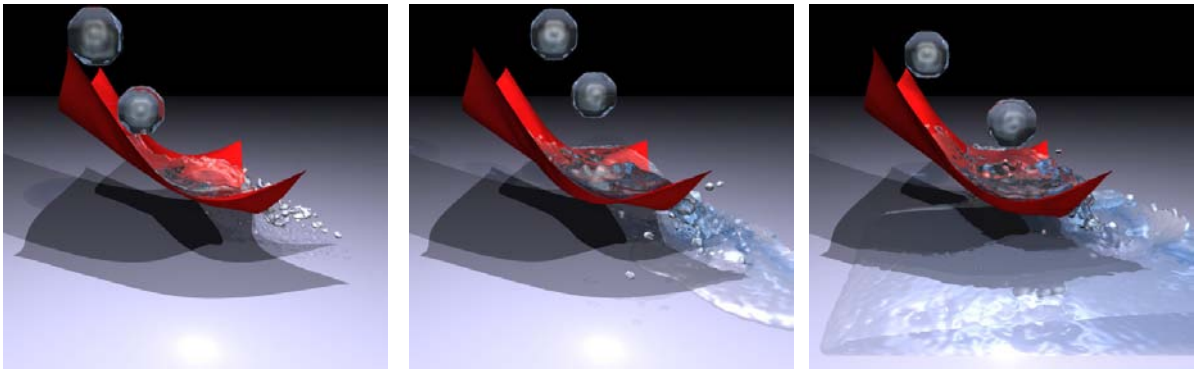

Figure 8: *Results of a simulation in which droplets are dropped onto a sheet of cloth.*

8. Conclusions

In this paper, we presented a method that can simulate the interaction between fluids and cloth in real-time. The interacting force is calculated by calculating an approximate distance between fluid particle and cloth. To improve the efficiency of the simulation, two grids are introduced. Then, we showed that the presented method can be entirely parallelized and accelerated using GPUs.

The interaction model presented in this paper is applicable to other particle-based simulations such as a granular material simulation by Distinct Element Method and a rigid simulation in which a rigid body is represented by a set of spheres. We accelerated the interaction between polygons and particles on GPUs, and this can also be applied to an elastic body simulation using tetrahedral meshes. Although all of the results shown in this paper are computed in real-time, the method is also able to accelerate an offline simulation.

References

[AIY*04] AMADA T., IMURA M., YASUMOTO Y., YAMABE Y., CHIHARA K.: Particle-based fluid simulation on gpu. In *2004 ACM Workshop on General-Purpose Computing on Graphics Processors* (2004).

[EMF02] ENRIGHT D., MARSCHNER S., FEDKIW R.: Animation and rendering of complex water surfaces. *ACM Transactions on Graphics 21* (2002), 721–728.

[FF01] FOSTER N., FEDKIW R.: Practical animation of liquids. In *Proc. of SIGGRAPH* (2001), pp. 15–22.

[FM96] FOSTER N., METAXAS D.: Realistic animation of liquids. *Graphical Models and Image Processing 58*, 5 (1996), 471–483.

[GHD03] GÉNEVAUX O., HABIBI A., DISCHLER J.: Simulating fluid-solid interaction. In *Graphics Interface* (2003), pp. 31–38.

[GRLM03] GOVINDARAJU N., REDON S., LIN M., MANOCHA D.: Cullide: Interactive collision detection between complex models in large environment. In *Proc. of ACM SIGGRAPH/Eurographics Symposium on Graphics Hardware* (2003), pp. 25–32.

[GSLF05] GUENDELMAN E., SELLE A., LOSASSO F., FEDKIW R.: Coupling water and smoke to thin deformable and rigid shells. *ACM Transactions on Graphics 24* (2005), 910–914.

[HBSL03] HARRIS M., BAXTER W., SCHEUERMANN T., LASTRA A.: Simulation of cloud dynamics on graphics hardware. In *Proc. of the SIGGRAPH / Eurographics Workshop on Graphics Hardware* (2003), pp. 92–101.

[HCSL02] HARRIS M., COOMBE G., SCHEUERMANN T., LASTRA A.: Physically-based visual simulation on graphics hardware. In *Proc. of ACM SIGGRAPH/Eurographics Symposium on Graphics Hardware* (2002), pp. 109–118.

[HKK07] HARADA T., KOSHIZUKA S., KAWAGUCHI Y.: Smoothed particle hydrodynamics on gpus. In *Proc. of Computer Graphics International, To Appear* (2007).

[KC05] KOLB A., CUNTZ N.: Dynamic particle coupling for gpu-based fluid simulation. In *Proc. of 18th Symposium on Simulation Technique* (2005), pp. 722–727.

[KFCO06] KLINGER B., FELDMAN B., CHENTANEZ N., O'BRIEN J.: Fluid animation with dynamic meshes. *ACM Transactions on Graphics 25* (2006), 820–825.

[KW06] KIPFER P., WESTERMANN R.: Realistic and interactive simulation of rivers. In *Proc. of the 2006 Conference on Graphics Interface* (2006), vol. 137, pp. 41–48.

[LC87] LORENSEN W., CLINE H.: Marthing cubes: A high resolution 3d surface construction algorithm. In *Proc. of the 14th Annual Conference on Computer Graphics and Interactive Techniques* (1987), pp. 163–169.

[LFWK03] LI W., FAN Z., WEI X., KAUFMAN A.: Gpu-based flow simulation with complex boundaries. *Tech-*

nical Report, 031105, Computer Science Department, SUNY at Stony Brook (2003).

[LLW04] LIU Y., LIU X., WU E.: Real-time 3d fluid simulation on gpu with complex obstacles. In *Proc. of the Computer Graphics and Applications, 12th Pacific Conference on (PG'04)* (2004), pp. 247–256.

[LSSF06] LOSASSO F., SHINAR T., SELLE A., FEDKIW R.: Multiple interacting liquids. *ACM Transactions on Graphics 25* (2006), 812–819.

[MCG03] MÜLLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. In *Proc. of SIGGRAPH Symposium on Computer Animation* (2003), pp. 154–159.

[Mis03] MISHRA B.: A review of computer simulation of tumbling mills by the discrete element method: Particontact mechanics. *International Journal of Mineral Processing 71*, 1 (2003), 73–93.

[MSKG05] MÜLLER M., SOLENTHALER B., KEISER R., GROSS M.: Particle-based fluid-fluid interaction. In *Proc. of Siggraph Symposium on Computer Animation* (2005), pp. 237–244.

[MST*04] MÜLLER M., SCHIRM S., TESCHNER M., HEIDELBERGER B., GROSS M.: Interaction of fluids with deformable solids. *Journal of Computer Animation and Virtual Worlds 15*, 3 (2004), 159–171.

[OLG*05] OWENS J. D., LUEBKE D., GOVINDARAJU N., HARRIS M., KRÜGER J., LEFOHN A. E., PURCELL T. J.: A survey of general-purpose computation on graphics hardware. *Eurographics 2005, State of the Art Reports* (2005), 21–51.

[Sta99] STAM J.: Stable fluids. In *Proc. of ACM SIGGRAPH 99* (1999), pp. 121–128.

[VSC01] VASSILEV T., SPANLANG B., CHRYSANTHOU Y.: Fast cloth animation on walking avatars. *Computer Graphics Forum 20*, 3 (2001), 260–267.

[WLMK04] WEI X., LI W., MUELLER K., KAUFMAN A.: The lattice-boltzmann method for simulating gaseous phenomena. *IEEE Transactions on Visualization and Computer Graphics 10*, 2 (2004), 164–176.