TOWARDS PRACTICLE REAL-TIME WATER SIMULATIONS: MULTIPHASE SMOOTHED PARTICLE HYDRODYNAMICS (M-SPH)

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER

ENGINEERING AND THE INSTITUTE

OF ENGINEERING AND SCIENCE

OF BİLKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Göktuğ F. Gökdoğan

September, 2008

ABSTRACT

TOWARDS PRACTICAL REAL-TIME WATER SIMULATIONS: MULTIPHASE SMOOTHED PARTICLE HYDRODYNAMICS (M-SPH)

Göktuğ F. Gökdoğan
M.S. in Computer Engineering
Supervisors: Prof. Dr. Bülent Özgüç and
Asst. Prof. Dr. Tolga Çapın
September, 2008

Simulation of water and other fluid phenomena have always been a popular topic in the computer graphics research area and many solutions provided in this topic covers many fluid simulation aspects. However, with the complex nature of physics of fluid dynamics, usually these solutions are not applicable to the real-time domain, especially interactive applications like computer games. The solutions that both target a realistic behavior and real-time CPU boundaries tend to solve the problem by utilizing Smoothed Particle Hydrodynamics (SPH) technique in the solution of Navier-Stokes equations. In this study, we introduce a novel approach for modeling of the water dynamics with multiple layers of SPH. This approach increases the level of detail in the constructed water surfaces while decreasing the required overall computation time. To achieve this, an extra SPH layer is introduced to use larger particles to fill most of the fluid volume which helps to simulate general fluid behavior in less numbers while utilizing other extra SPH layers with small particles to fill up in-betweens for finer detail in water surfaces. The performance gain can be up to several magnitudes with the increase of the water size while maintaining visually similar or more appealing results.

Keywords: Natural phenomena, Computational Fluid Dynamics, Navier-Stokes Equations, Physically-based modeling, water animation, real-time fluid simulation, Smoothed Particle Hydrodynamics.

ÖZET

UYGULANABİLİR GERÇEK-ZAMANLI SU SİMÜLASYONLARINA DOĞRU: ÇOK-FAZLI YUMUŞATILMIŞ PARÇAÇIK HİDRODİNAMİĞİ (M-SPH)

Göktuğ F. Gökdoğan
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Yöneticileri: Prof. Dr. Bülent Özgüç ve
Yrd. Doç. Dr. Tolga Çapın
Eylül, 2008

Su ve diğer akışkanların benzetimi, bilgisayar grafiğinde hep popüler bir konu olmuştur ve bu konuda akışkan simülasyonunun değişik yönlerini içeren pek çok çözüm sunulmuştur. Fakat, bu çözümlerin çoğu akışkanlar dinamiğinin karmaşık doğasından dolayı, gerçek zamanlı konulara, özellikle de bilgisayar oyunları gibi kullanıcı etkileşimi olan uygulamalara uygulanabilir nitelikte değildir. Gerçekçi bir davranışın yanında gerçek zamanlı koşum kısıtlarına da uymayı hedefleyen çözümler genellikle Navier-Stokes denklemlerini Yumuşatılmış Parçacık Hidrodinamiği (*Smoothed Particle Hydrodynamics* - *SPH*) tekniğini kullanarak çözme eğilimindedirler. Biz bu çalışmada, su dinamiğini modellemek için birden çok SPH katmanı kullanarak yeni bir yaklaşım ortaya koyuyoruz. Bu yaklaşım, toplam hesaplama zamanını azaltırken, su yüzeyindeki detay seviyesini de artırmaktadır. Bunun için, su yüzeyini daha detaylı göstermek üzere küçük parçacıklar kullanan SPH katmanları kullanılırken, suyun geriye kalan büyük kısmına genel akışkan davranışını koruyacak şekilde daha az sayıda ancak daha büyük parçacıklar kullanan katmanlar uygulanmıştır. Bu şekilde, su miktarı arttıkça katlanan performans kazanımları sağlanırken, görsel açıdan benzer veya daha gerçekçi sonuçların alınması başarılmıştır.

Anahtar Sözcükler: Doğal Fenomenler, Hesaplamalı Akışkanlar Dinamiği, Navier-Stokes Denklemleri, Fizik Tabanlı Modelleme, Su Animasyonu, Gerçek Zamanlı Akışkan Simülasyonu, Yumuşatılmış Parçacık Hidrodinamiği.

Acknowledgement

I wish to express my gratitude to Prof. Dr. Bülent Özgüç and Asst. Prof. Dr. Tolga Çapın for giving me the opportunity to achieve this thesis, and for their supervision and support. I am also grateful to Assoc. Prof. Dr. Uğur Güdükbay, Assoc. Prof. Dr. Veysi İşler and Asst. Prof. Dr. Murat Karamüftüoğlu for their valuable comments on this thesis.

Finally, I would like to thank my family for their endless support in my education and increasing my motivation,

And my wife, for her love and sacrifices that made this possible, and being near with me all the time that gave me the power to pursue this research. Nothing is sufficient to pay back all that I owe her.

Contents

List of Figures

List of Tables

Eşime...

Chapter 1

Introduction

Computer graphics artifacts have a serious place in our lives in many ways through computer games, animated movies, defense and medical applications etc. The variety of usage of computer graphics not only serves the commercial and academic purposes, but also contributes and accelerates the technical advances devoted to ease life directly and indirectly. Realistic representation of the natural phenomena, especially representation of liquids like water, is one of the most interesting topics in computer graphics. Although human interaction with liquids, such as water, is high and the behavior, physics and structure of them have been well known for decades, their simulation is still a great challenge for computer graphics. According to Jeffrey Katzenberg, who is the DreamWorks SKG principal and producer of well-known animation movie series *Shrek*, pouring of milk into a glass is the single hardest shot in *Shrek* [Enr02].

Extensive research has been performed for simulation of fluids addressing different concerns because of the challenges and attraction of the topic. Some of the works concentrate on the realism of fluid simulation; on the other side, some concentrate on real-time modeling and simulation of fluids, and some concentrate on controlling the behavior of fluids for animation with an acceptable visual realism. If the realism of simulation is important, the fluid's physical properties have to be modeled in detail, the interaction of the fluid with its environment, like air, has to be well defined and modeled, and extra computations need to be considered to visualize special animation effects like "splashing". Although works which aim at the photo-realism of the fluid simulation end up with visually plausible results, the detailed physical modeling and rendering costs are very high, such that, the computation of a physical effect can take many hours even in a farm of workstations dedicated for this job.

The realism of the simulation is a very big concern for animated movies, but for a real-time application, like a 3D computer game, there is a big trade-off for the realism of the simulation considering that the application must run at 40-60 frames per second at constant rate on a single computer. Moreover, this rate is not only devoted to computation of a physical effect but also includes rendering and character animation tasks. To satisfy such a constraint, an application needs to run a fast algorithm to consume as little computational power as possible. Physics simulation tasks can now be made on GPUs (Graphical Processing Units) or by more specialized devices such as PPUs (Physics Processing Units) instead of CPUs; however, consuming less computational power is still the most important concern for real-time applications. Besides, since the available memory is restricted with the target platform's memory, which also has to be shared with other tasks in real-time computer games, the memory consumption of the application need to be low unlike the off-line computations [Bri06].

As a very useful set of equations to describe the physics of fluids like liquids and gases, Navier-Stokes equations are utilized in many liquid simulation applications for realistic visualization, especially in animated movies. However, as a result of the constraints described above, these equations are not used to be very practical for real-time liquid simulations and several methods are introduced for simulation of liquids in real-time such as Procedural Water, Heightfield Approximations, Particle Systems etc.. Procedural Water method does not simulate the cause of physical effect, but simulates the physical effect itself. For example, it uses sine waves at different amplitudes and directions to simulate surface of an ocean. This technique gives the animator a high degree of control of the animation, but the boundary interaction of the water is very difficult to simulate. Heightfield Approximations method is used for simulation of a lake or ocean surface. The surface is represented with a 2D heightfield and a 2D wave equation is used for animation. Since 2D heightfields are used, the breaking waves cannot be simulated. Particle Systems method is used to simulate a small body of water like puddles, runnels or splashing fluids; and is usually used with Heightfield Approximations method for advanced effects. [Bri06]

Apart from the above mentioned methods, there are some improvements introduced for solving the Navier-Stokes equations in real-time simulations. The Smoothed Particle Hydrodynamics (SPH) method is used to solve the Navier-Stokes equations in Müller's research [Mül03]. However, simulation of pouring water into a glass works at 5 frames per second, which is still far from optimum and needs further optimizations.

In our study, we make an additional advancement for real-time modeling and simulation of fluid behavior by using SPH techniques to solve the Navier-Stokes equations. We introduce the concept of layers in SPH methodology that increases the level of detail and decreases the computational power consumed. Our new method helps utilization of smaller particles at near surface areas to increase the level of detail and bigger particles that require less computation effort for the rest of the volume.

The organization of the rest of the thesis is as follows. Chapter 2 provides a general background on the topic by introducing the physics behind fluid phenomena and gives some of the state of the art solutions on the subject. Chapter 3 describes

Smoothed Particle Hydrodynamics technique and its application to Navier-Stokes equations. Chapter 4 introduces our proposed approach, Multiphase SPH, to model water dynamics. Chapter 5 evaluates the approach and gives the visual results. Chapter 6 concludes the thesis with possible future research directions.

Chapter 2

Background

2.1 Physics behind Water

Water is an example of complex natural phenomena where lots of physics get involved including computational fluid dynamics. We will first investigate the basic physical properties of water and then introduce physical basis for its flow in time with the help of computational fluid dynamics.

2.1.1 Physical Properties of Water

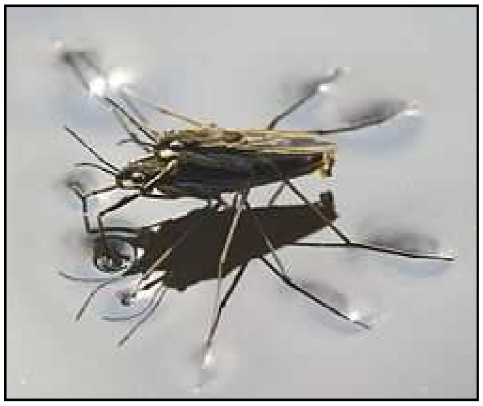
Density: Density is defined as a substance's mass (m) to volume (v) ratio. It is formulated as

$$p = \frac{m}{v} \tag{1}$$

Water's density is measured as 0.9999720 which reaches to a maximum at +4 degrees Celsius [Den08].

Viscosity: Viscosity is the fluid resistance against flowing, or its stickiness. Liquids with high viscosity, like pitch are thick liquids and resistant to stress. Liquids with low viscosity, like water, are thin liquids and are not resistive to stress. According to the Newton principle, shear stress *T*, between fluid layers is proportional to velocity gradient with viscosity multiplier. This principle is generally applicable for fluids like water and most of the gases. Viscosity of liquids is not pressure dependent and it is inversely proportional to temperature. For example, water's viscosity is 1.79 cP (centipoises) at 0 degrees Celsius and 0.28 cP at 100 degrees Celsius [Vis08].

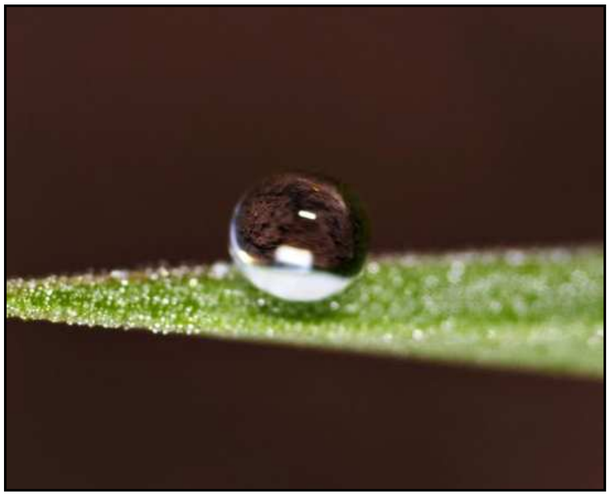
Surface Tension: Surface tension is a result of the affinity between liquid molecules. A liquid molecule is pulled in all directions by the neighboring molecules which results in a zero net force. However, a molecule which is at the surface, is pulled inwards by the neighbor molecules which are at a deeper position but not pulled very well by other substances at the interaction points resulting in a non-zero net force. Since the surface molecules are pulled towards the inside, they are squeezed to a more resistant volume that constructs a smaller surface area. As a result, the surface of the liquid behaves like a stretched membrane or an elastic sheet which we call as surface tension. By the help of surface tension, insects like water strider can walk on the water and small objects can float on the water. Figure 2.1 shows an insect stand on the water without sinking:



(Downloaded from: http://en.wikipedia.org/wiki/Image:Wasserläufer_bei_der_Paarung_crop.jpg, Author: Markus Gayda)

Figure 2.1: Water strider

Hydrophobe: Hydrophobicity means physical property of being repelled by water. These kinds of molecules are non-polar molecules, which cannot be dissolved in water, like oil and they have a high contact angle with water. Water is polarized and can construct hydrogen bonds internally. But hydrophobic substances are not polarized and they cannot construct hydrogen bonds. Because of this, water repels the hydrophobic substances to be able to bond with itself. Figure 2.2 shows water drops on hydrophobic leaves.



(Downloaded from: http://en.wikipedia.org/wiki/Image:Dew_2.jpg, Author: Michael Apel)

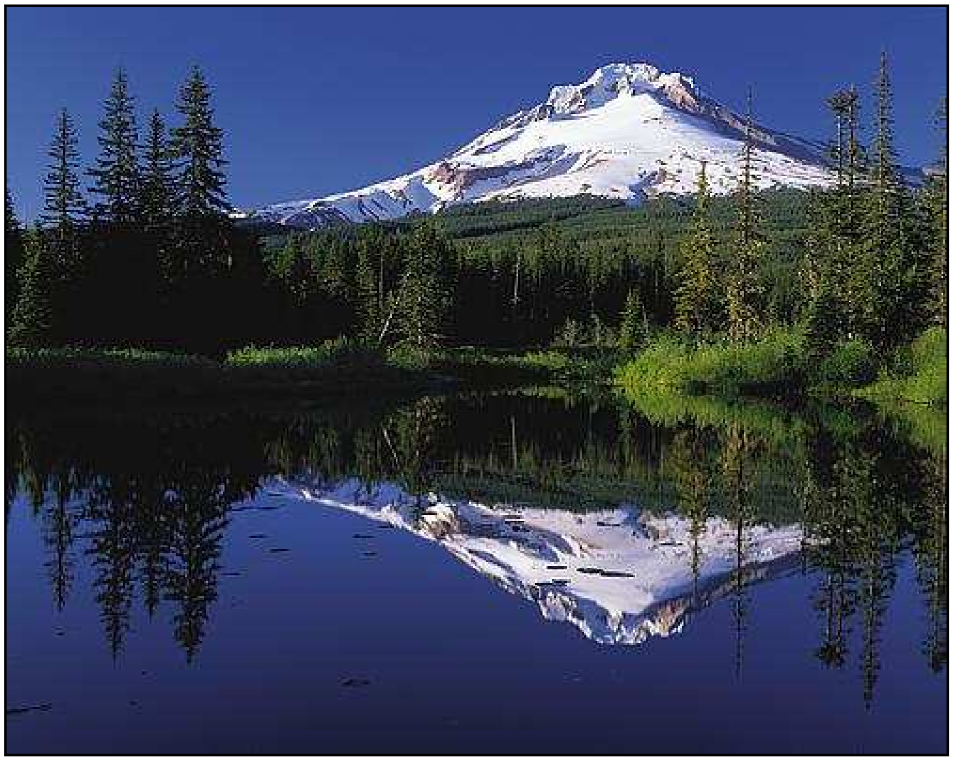
Figure 2.2: Water beading on leaf

Hydrophile: Hydrophile means a physical property of constructing hydrogen bonds with water temporarily. These kinds of substances are polar and electrically polarized substances which can be dissolved in water unlike the hydrophobic substances. Paper towel made from cellulose is an example for hydrophilic substances.

Compressibility: Compressibility stands for the change of fluid and solid volume due to the pressure change that is also dependent to temperature. Since non-gas substances like water have very low compressibility, they are also referred as incompressible. For example, in a deep ocean where the depth is 4000 meters and the pressure is 40 000 000 Pa, the volume decrease is just 1.8% [Wat08].

2.1.2 Optical Properties of Water

Reflection: Reflection is the direction change of light or sound caused by a collision to a medium surface. It is categorized as specular reflection when a ray is reflected to a single direction and as diffuse reflection where a ray is reflected to several directions. Water generates specular reflections. Figure 2.3 shows reflection of a mountain on water:



(Downloaded from: http://en.wikipedia.org/wiki/Image:MtHood_TrilliumLake.jpg, Author: Oregon's Mt. Hood Territory)

Figure 2.3: A mountain’s reflection on water

Refraction: Refraction is described as change of the direction of a wave when its speed is changed generally due to moving from one medium into another. According to the Snell's law, angle of incidence is related to the angle of refraction as follows:

$$\frac{sin(A_1)}{sin(A_2)} = \frac{v_1}{v_2} = \frac{n_2}{n_1} \tag{2}$$

where,

v is the velocity in respective environments

A is the angle between normal planes in respective environments

n is the refraction indices in respective environments[Ref08]

The most popular example of refraction can be seen by looking at a glass of water where a straw is immersed. Since the air's and water's refractive indices are different, the straw will seem as if it is bending on the surface of water.

2.1.3 Fluid Dynamics

The animation of water is made possible by approximating the physical motion characteristics (dynamics) of water in a computational environment. The dynamics of water is researched under the subject of fluid dynamics.

In general, fluid dynamics is concerned with flow of fluid in any form (liquid and gases). It has a wide range of applications including calculation of force and moments on aircraft, determining mass flow rates on pipes and predicting weather patterns [Flu08].

According to fluid dynamics, water conforms to the law of conservation of mass, conservation of linear momentum and conservation of energy (in terms of classical physics).

Conservation of mass law states that in a closed system the mass will remain constant in time no matter what happens/acting inside the system. Conservation of momentum guarantees that total momentum in a closed system without an act of external force will remain constant. So we can expect the momentum can be transferred between objects as long as the total momentum does not change. With conservation of energy, the system will maintain the same amount energy which can possibly be redistributed in different forms.

The flow of fluid is described by some set of non-linear differential equations known as Navier- Stokes equations that dictates these conservations. It is the fundamental basis of computational fluid dynamics. Simulation of water can be considered as solving of these equations (or heavily simplified versions of these equations) with respect to the time.

2.1.4 Lagrangian and Eulerian Viewpoints

The Lagrangian and Eulerian methods, named after French mathematician Lagrange and Swiss mathematician Euler, are the most common techniques to track fluid flows or deformable solid moves. The core of Lagrangian viewpoint is thinking the flow or continuum as a particle system, and each point in the flow or continuum as a particle. The method is independent from the particle size such that, the particles can be considered as molecules or big continuum parcels. The Lagrangian method is good to simulate solids as a discrete set of particles in a mesh. The core of Eulerian viewpoint is looking at fixed points in space and tracking the changes of the fluid properties like velocity, temperature and density at that point. This method is generally used for fluids by using a fixed grid in space through which fluid passes. A flowing fluid contributes to the resultant quantity of a point when it passes through that point. For example when a hot fluid passes through a fixed point, and then a cold fluid passes through the same point, the temperature of the point decreases, although the single particle temperatures of the fluid are constant.

A good example to differentiate between these two viewpoints is to consider the use of these methods for weather forecasting. If we are using Lagrangian viewpoint, we track the air properties like temperature, pressure, humidity etc. while we are in a balloon and moving with the wind. If we are using the Eulerian method, we stand on the ground and track the air properties which are flowing past at the point we stand [Bri06].

Although the Lagrangian method seems to be more suitable and simple intuitively and Eulerian method seems to be complicated, working with spatial derivatives like viscosity and approximating the spatial derivatives on a fixed Eulerian grid are easier in the Eulerian method. Consequently, many researches use Lagrangian and Eulerian methods side by side [Bri06].

2.1.5 Computational Fluid Dynamics Solutions

In Computational Fluid Dynamics, treating the continuous flow in a discretized way is the main concern. One way for this is constructing small meshes or grids and applying the algorithm to solve the motion equations on them which can be Euler equations or Navier-Stokes equations. A solution that does not use grid-based method is SPH method, which is a Lagrangian method to solve Navier-Stokes equations for fluid flow (details given in Chapter 3). In addition to SPH, there are Spectral methods, which are used to solve partial differential equations that describe fluid flow or sound propagation like physical processes by using Fast Fourier transformations. Besides, there are Lattice Boltzmann methods, which simulate the Newtonian fluids with collision models without using Navier-Stokes equations. Those methods use Boltzmann equations instead, which are not interested in conservation of mass, momentum or energy and instead use imaginary particles which perform consecutive propagation and collision processes.

2.1.6 Navier-Stokes Equations

In an ideal fluid, the Navier–Stokes equation creates a relation between the acceleration of fluid and the gradient of pressure. Instead of explicitly dictating a position, it dictates the velocity while defining the flow.

The most general form of the equations is:

$$\rho\left(\frac{\partial v}{\partial t}+ v \cdot \nabla v\right) = -\nabla p + \nabla \cdot T + f \tag{3}$$

where,

ρ is the density

v is the velocity

p is the pressure

T is the stress tensor

f is the force per unit volume

In incompressible flows (which is our interest), the $\nabla \cdot T$ term describes the viscous forces which are defined by $\mu\nabla^2 v$ (See [Der08] for detailed derivation). So we can replace it immediately:

$$\rho\left(\frac{\partial v}{\partial t} + v \cdot \nabla v\right) = -\nabla p + \mu\nabla^2 v + f \tag{4}$$

Written in the substantial form:

$$\rho\frac{Dv}{Dt} = -\nabla p + \mu\nabla^2 v + f \tag{5}$$

where $\frac{Dv}{Dt}$ denotes the acceleration including possible convective effects ($v \cdot \nabla v$), for example; acceleration due to a nozzle. While describing the fluid flow with the Eqn. (5), we actually define the conservation of momentum. With mass continuity equation we formalize our other assumption, conservation of mass:

$$\frac{\partial\rho}{\partial t} + \nabla \cdot (\rho v) = 0 \tag{6}$$

In a Lagrangian viewpoint (if we move with the material), the substantial derivative form of the above equation is more useful:

$$\frac{D\rho}{Dt} + \rho(\nabla \cdot v) = 0 \tag{7}$$

With our assumption of incompressible flow while moving with the material, the density should not change. That makes the first part of the equation equal to zero:

$$\frac{D\rho}{Dt} = 0 \quad \rightarrow \quad \rho(\nabla \cdot v) = 0 \quad \rightarrow \quad \nabla \cdot v = 0 \tag{8}$$

2.2 Background Work

Foster and Fedkiw described a general model to animate fluids in a 3D environment [Fos01]. This model is a combination of an extension of semi-Lagrangian method and a new method to calculate the flow of fluids around objects. The physically based animations usually use direct numerical simulations to be realistic. In this kind of techniques, initial and boundary conditions are specified and the simulation runs freely with a very limited effect of the animator. However, by the help of combined method presented in [Fos01], not only the numerical techniques are used to achieve the realism but also the local and global motion of the fluid is controlled to match the behavior of the fluid aimed to be created. The method is applicable for animating diluted liquids like water and dense liquids like thick mud.

To achieve this extensive system for modeling and animating the fluids, firstly Navier-Stokes equations for incompressible flow of fluids are solved by using a semi-Lagrangian method. As a result, the fluid is modeled as velocity and pressure couple dynamic fields, and the fluid motion is determined by generating these fields over time in a rectangular grid of voxels due to effects of viscosity, convection, density, pressure and gravity. But these methods generally cause mass dissipation, which is not a problem for gases, but is a problem for liquids since it has visually disturbing effects. The mass dissipation is prevented by a new method which is called the hybrid surface model that combines the inertialess particles and an implicit surface which is called level set. Particle development is a Lagrangian method which suffers from visual artifacts in small particles and the level set is an Eulerian method which does not suffer from it but has a difficulty resolving features with sharp corners. When a combination of these methods is used, the inertialess particles allow the liquids to splash freely and the level set prevents the mass dissipation. Another part of the technique described in this paper involves considering the moving polygonal objects' effects in the liquid. A new technique is developed for this purpose, which enables low frequency control over the fluid volume and calculates the fluid behavior. The control of fluids involves the moving object mechanism such that, instead of moving the polygons themselves, fake surfaces are introduced which have normals and velocities directed to the direction which the liquid is requested to go. The fluid behavior is calculated by allowing the liquid to be pushed along by the object and in addition by allowing to slide freely around the object.

Enright et al. proposed a new method for photo-realistic simulation of 3D water effects like pouring water into glass and ocean wave breaking [Enr02]. To achieve a more photo-realistic water surface, a thickened front tracking technique, which is called “particle level set”, is introduced. This hybrid method uses the mass-less particles and dynamic implicit surfaces. This proposed model is an improvement of [Fos01], and it provides much more realistic surface behavior by modeling surfaces rather than the volume. The model in [Fos01] is weak for the non-liquid surfaces, such as air, because it loses air volume and causes the liquid to gain the lost air volume. As a result of this erroneous volume shift, dynamic splashing effects are destroyed in the simulation. The proposed model runs at 11 minutes per frame comparing to previous model which is 7 minutes per frame. The rendering takes 15 minutes per frame but no hardware configuration was given.

To handle a degree of control on surface motion for generating a wind blown appearance or for making the water to settle quickly, a new velocity extrapolation method has been introduced.

As a rendering technique, photon mapping has been chosen, because of its simplicity and for avoiding the distracting noise which happens in pure path sampling algorithms (e.g., ray tracing). The rendering technique is described in [Jen01].

Müller et al. presented a Smoothed Particle Hydrodynamics (SPH) based physical model [Mül03]. Although the SPH method has previously been used to simulate fire and other gaseous phenomena [Sta95] and afterward is used to animate highly deformable bodies [Des96], Müller et al. improved it to simulate random fluid motion. It uses Navier-Stokes equations on particles by deriving the force density fields and by adding surface tension modeling. In addition, to introduce the particles' effects on each other, smoothing kernel method is applied. Apart from the Eulerian methods, this particle based method can be administrated without the mass conservation equations and convection terms. In this way the complexity of the simulation is decreased. To visualize and track the physical model described, marching cubes method and point splatting methods are proposed to construct a surface from the particles. At performance point of view, this animation runs at 20 frames per second on a 1.8 GHz Intel® Pentium® IV PC with a GeForce® 4 graphics card when sampled with 2200 particles.

Similar to [Mül03], Claver et al. have proposed a particle-based model in which SPH technique is used, except that, extensions and modifications are made to simulate more realistic viscoelastic fluids such as paint and mud and their splashing effects on moving solids, without worrying about Navier-Stokes [Cla05]. To perform a more realistic simulation for viscoelastic liquids, each particle pair is modeled with varying rest length springs between them. According to the material behavior of the liquid, the spring rest length value is adjusted.

Double density relaxation procedure, which is an extension of the SPH method, is used for enforcing incompressibility and particle anti clustering. For each particle, a local density is calculated by summing weighted contributions of the particle's neighbors. If the calculated density is lower than the rest density, the particle will be pulled and if it is greater than the rest density, the particle is pushed proportional to the linear kernel function. This calculated displacement of the particle due to pulls and pushes directly changes the estimated position of a particle in one direction and the other particle which has attached with a spring to the opposite direction according to the action-reaction principle. However, since a particle can pull the small neighbor particles strongly, particle clustering can be observed and as a precaution of this clustering effect, a distance dependent repelling force, which is named as near-pressure, is used. With the help of this repelling force, the neighbor particle is pushed proportional to a quadratic kernel. For the particles near surface, a non-zero net force appears towards the fluid which will perform the surface tension. Additionally, viscoelasticity behavior is simulated by combination of elasticity effect, which is handled by inserting springs between particles, plasticity effect, which is handled by modifying the spring rest lengths, and viscosity effect, which is handled by smoothing the velocity field. The described fluid simulation can be integrated into a rigid-body system to visualize floating objects and liquid sticking on surfaces. Collision detection and stickiness effects are introduced to the system to manage these object interactions. As a rendering technique, marching cube algorithm is utilized.

In [Mül05], the interactions between different fluid types like air-water, water-wax have been analyzed. To model the fluid-fluid interactions, a Smoothed Particle Hydrodynamics based new technique is presented instead of an Eulerian grid based method, since simulation of multiple fluids and simulation of the different phases of the fluids is very difficult in Eulerian methods. In the introduced method, different fluids with different phases are represented as individual particles, such that each of the particles has its own attribute values. In this way, the particles of different fluids at different phases can be mixed randomly and they can be created or destroyed dynamically. This technique is used to simulate the trapped air, multiple fluids and phase transitions of fluids. For simulation of multiple fluids, regular SPH method is extended to simulate the different fluids' interactions with each other. In the conventional SPH method, particle attributes like mass, rest density, viscosity coefficient are same for all particles. On the contrary to the SPH method, Müller et al. stored these properties individually for each particle and additional attributes are introduced. Color attribute is introduced for simulation of interface and surface tension, and temperature attribute is introduced for simulation of effects like phase transition.

Buoyancy interaction is simulated by using the individual rest density of each particle. When several fluids with different rest densities are mixed, a pressure gradient, which enables diluted particles to go over the nearby dense particles. The immiscible liquid interactions are simulated by using the interface tension between polar and non-polar fluids like water and oil. The diffusion interaction is simulated by modeling the temperature of unique particles according to a diffusion equation. According to the ideal gas law, temperature and density are inversely proportional and hence, the temperature influences the rest density.

For simulation of trapped air, air particles have been introduced to the model, contrary to the standard SPH model. However, simulation of all air particles including the air outside of the liquid requires huge number of particles. To do this, air particles are generated dynamically only at the places where bubbles are formed and they are destroyed when they are isolated from the liquid. This technique increases the realism of pouring water into a glass by the help of buoyancy interaction.

For simulation of phase transitions like boiling water, the particle's types and densities are changed dynamically with respect to their temperatures. At first, random cavitation points are chosen in a glass of water. When the water particles near the cavitations reach 100 degrees, they are turned into air particles. Since the air particles have less density then the water particles, they arise through the surface.

Recently, Hong et al. presented an adaptive approach for simulating incompressible fluids using the hybrid FLIP method [Hon08]. In this method, a Marker-and-Cell (MAC) grid is constructed around the particles and then particles' velocities are transferred to this grid. Forces due to external effects and diffusion are applied and the final acceleration is integrated to find the velocity change. A Poisson equation is solved for finding the pressure field and this field's gradient is used to correct the velocities. Finally the velocities are reflected back to particles in order to find the new positions.

The merging and splitting decision is based on 2 terms; the Reynolds number and the distance from surface. The Reynolds number is the ratio of inertial forces to viscous forces and identifies the highly deformable areas. The distance from the surface is used to divide the fluid into layers and is calculated by the help of Fast Sweeping Method. The particles that enter to the closest layer to surface are automatically split into smaller ones. In other layers, a split or merge decision is made using the Reynolds number.

Experiments show that 25% percent performance increase is achieved in simulation when only the merging is performed. If both merging and splitting are performed (adaptive mode) the speed is decreased by 10% percent in favor of a finer representation of fluid.

Chapter 3

Smoothed Particle Hydrodynamics

Smoothed Particle Hydrodynamics (SPH) is a computational method for solving fluid flow problems. It is based on integral interpolants ([Luc77],[Gin77]). The method was first used in astrophysical problems for modeling hydrodynamic flows in three dimensional space [GRL03].

The main motivation behind Smoothed Particle Hydrodynamics is encoded in its name:

The first term *SMOOTHED* states the approximation of attributes using the weighted average over the neighboring particles.

The second term *PARTICLE* represents the smallest computational term in the calculation.

The third term *HYDRODYNAMICS* defines application domain of the method which is the hydrodynamics problems.

Smoothed Particle Hydrodynamics is categorized under Lagrangian methods as the coordinates move with the fluid. Instead of viewing the fluid system consisting of regular, distributed homogeneous points of grid, SPH uses particles as sample points which form an irregular computation area. In other words, by using particles instead of sampling the space by a uniform grid, it works on a set of discrete fluid elements. As a result, the system consists of less discrete particles instead of grid points. In this way, computation is dramatically simplified computation, which aids us to simulate water dynamics in real-time.

Each particle in Smoothed Particle Hydrodynamics differs from regular particles (in the concept of particle systems) with their continuous and smooth representation of quantities. In a regular particle, the particle itself represents an existence; it has its own volume, mass, color. However, for a SPH particle, there is no concept of own volume, its existence represents some form of computation center. In this way, it provides a continuous instead of a discrete existence. To provide this continuity, each attribute value of the particle is smoothly interpolated over the area in effect of particle. This interpolation function is called the Smoothing Kernel (W). The Smoothing Kernel interpolates the value of a given attribute x_i for a given location with a distance of r to the particle. As the function only depends on the distance, the function is naturally symmetric around the particle. Gaussian function and cubic spline are only a few of possible smoothing kernels. The effective area of the particle is determined by the smoothing kernel and the radius of the area is called the smoothing length (h). To calculate a physical quantity at any point in space we can sum up the weighted contributions of all particles that reside in the radius h of the point. We can compare the smoothing kernel based quantity calculation with interpolations in grid analogy:

In a regular grid-based sampling, for any point P we know where the grid cell P resides. In a grid cell we know the value of the quantity in each corner of the cell. So we can approximate the quantities value at point P by interpolating the values at the corners.

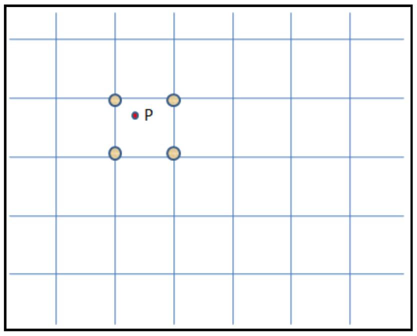


Figure 3.1: Grid approximation

In SPH based sampling, for any point P we know the particles that fall into the radius r of the particle. As we know the values of quantities in each particle that fall into the radius, we can calculate the quantity at P by interpolating the quantities using the interpolation function of the SPH, i.e., the smoothing kernel.

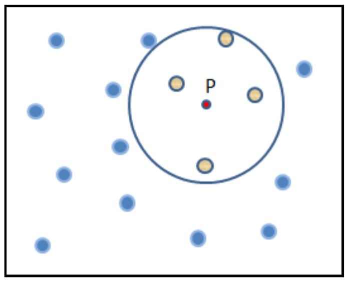


Figure 3.2: SPH based approximation

This quantity evaluation method in Smoothed Particle Hydrodynamics can be formalized with the following equation:

$$A(r) = \sum_j m_j \frac{A_j}{\rho_j} W(|r - r_j|, h) \tag{9}$$

where:

A is the quantity that the evaluation is performed for

r is the evaluation positionm_j is the mass associated with particle j

A_j is the value of scalar quantity at particle j

r_j is the position of particle j

ρ_j is the density associated with particle j

W is the smoothing kernel

h is the smoothing length

For example, the temperature at a location can be evaluated by:

$$T(r) = \sum_j m_j \frac{T_j}{\rho_j} W(|r - r_j|, h) \tag{10}$$

From a mathematical view point, the summation will include all particles of the system, but from an implementation viewpoint, the summation will include only the particles in smoothing length radius of particle in question; as the other particles will have exactly zero contribution to the calculation proved by the definition of smoothing length.

An important property of the formula appears when we evaluate the gradient of a scalar quantity. By using integration by parts, the ∇ operator shifts from the scalar quantity to the smoothing kernel. Therefore the gradient only affects the smoothing kernel. By this property we can easily calculate the gradient of a quantity by:

$$\nabla A(r) = \sum_j m_j \frac{A_j}{\rho_j} \nabla W(|r - r_j|, h) \tag{11}$$

Similarly the Laplacian can be evaluated by:

$$\nabla^2 A(r) = \sum_j m_j \frac{A_j}{\rho_j} \nabla^2 W(|r - r_j|, h) \tag{12}$$

These properties are especially important for simulation to take place in real-time because the gradient and the Laplace of the smoothing kernel can be constructed by hand and used throughout the simulation.

3.1 Using SPH to Solve Navier-Stokes Equations

As stated earlier in Section 2.1.6 Navier-Stokes equations are derived by the conservation of mass and conservation of momentum. Recall that, these equations are:

$$\rho \frac{Dv}{Dt} = -\nabla p + \mu \nabla^2 v + f \tag{5}$$

$$\frac{D\rho}{Dt} + \rho(\nabla \cdot v) = 0 \tag{7}$$

The second equation is used to conserve mass but as the fluid constructed as particles and the masses of particles do not change in time it can be immediately eliminated.

We can transform the first equation to an acceleration formula by pulling the density term to the right-hand side of the equation:

$$\frac{Dv}{Dt} = \frac{-\nabla p + \mu \nabla^2 v + f}{\rho} = a \tag{13}$$

Thus, we need to evaluate gradient of pressure ($-\nabla p$), viscosity effect ($\nabla^2 v$) and external forces (f) at each particle center in order to calculate acceleration at that particle.

The final term f is usually gravity or user effect and does not need a further calculation.

The gradient of the pressure at any particle center can be evaluated by applying Eqn. (11):

$$-\nabla p(r_i) = -\sum_j m_j \frac{p_j}{\rho_j} \nabla W(|r_i - r_j|, h) \tag{14}$$

To solve Eqn. (14), pressures at each particle point need to be calculated. [Mül03] suggests the following equation:

$$p = k(\rho - \rho_0) \tag{15}$$

Eqn. (15) needs densities to be evaluated for particles. This is a simple application of Eqn. (9) similar to temperature example in Eqn. (10):

$$\rho_i = \sum_j m_j \frac{\rho_j}{\rho_j} W(|r_i - r_j|, h) \tag{16}$$

which can be simplified to:

$$\rho_i = \sum_j m_j W(|r_i - r_j|, h) \tag{17}$$

Additionally [Mül03] modifies the Eqn. (14) in order to make symmetric pressure forces generated between each particle.

$$-\nabla p(r_i) = \sum_j m_j \frac{p_i + p_j}{2p_j} \nabla W(r_i - r_j, h) \tag{18}$$

Although the intention was symmetry in the modification, unfortunately it can be easily seen that the resultant equation does not produce symmetric forces due to possible differences in densities of the particles. We will come back to this later in our solution in Chapter 4.

The second term that needs to be evaluated is ($\nabla^2 v$):

$$\mu \nabla^2 v(r_i) = \mu \sum_j m_j \frac{v_j}{\rho_j} \nabla^2 W(r_i - r_j, h) \tag{19}$$

Similar to pressure forces, an attempt to symmetrize the viscosity effect yields:

$$\mu \nabla^2 v(r_i) = \mu \sum_j m_j \frac{v_j - v_i}{\rho_j} \nabla^2 W(r_i - r_j, h) \tag{20}$$

After each term is calculated, using Eqn. (13), the particles' positions can be calculated with simple Verlet integration.

Chapter 4

Multiphase SPH (M-SPH)

As stated in Chapter 3, the smallest computation unit in SPH is the particle. If we increase the number of particles per volume, then the approximation errors will decrease and the surface detail will increase, which will all result in more realistic simulation. But with the increase in the number of particles, the computation requirements will increase, and this will degrade the performance of the simulation. The number of particles per volume is directly proportional to mass quantity of each particle. Therefore, particle mass value is our key to simulation granularity.

In simple SPH, the mass for each particle is considered to be equal. Therefore, the calculation is in the same detail throughout the volume of the fluid. As most of the fluids are nearly transparent, the most appealing area in the eye of the observer is the reflection and refraction applied surface of the water. The particles that are most effective in the construction of this appealing area are the particles that are most close to the surface (Figure 4.1). If we are simulating a pool of water then we spend most of the computation time for simulating the particles that are least effective in the visualization since just a small portion of particles are close to the surface.

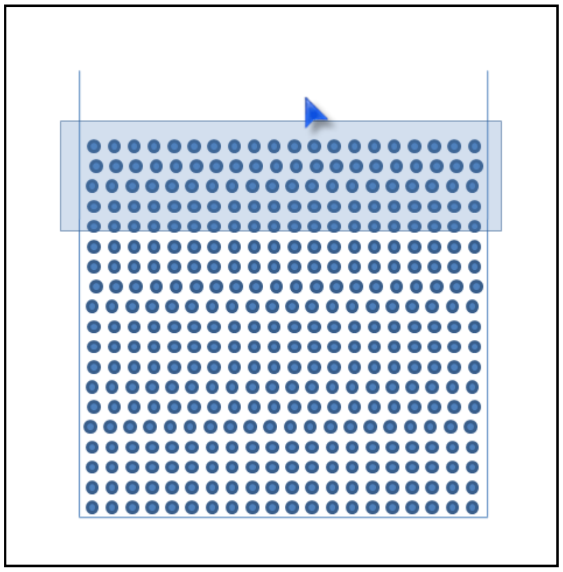


Figure 4.1: A pool of water

The proposed Multiphase Smoothed Particle Hydrodynamics method considers the water as a construction of several SPH layers. Each layer provides a different level of resolution and behavior for the simulation of water. These layers are combined with traditional SPH approximations and provide an optimized model for simulating fluid behaviors. While lower resolution layers keep the general fluid flow behavior, the higher resolution layers provide a finer level of detail for more complex surfaces.

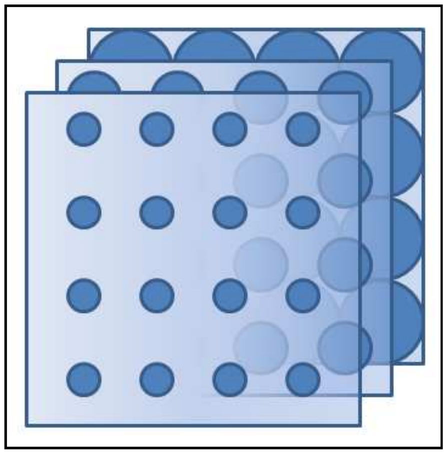


Figure 4.2: M-SPH layers

In conceptual level, we can consider each layer as a different SPH simulation. Like regular SPH, each layer has its own:

- mass per particle,
- smoothing length
- smoothing kernel (optionally)

Therefore at any point in space, we can calculate the density, velocity, any quantity for that layer according to regular SPH formulations. Nevertheless, considering the water itself, we need to take each layers contribution into account.

Taking multiple layers into account, we can calculate any scalar quantity in water space by adding each layers contribution:

$$A(r) = \sum_j m_j \frac{A_j}{\rho_j} W(|r - r_j|, h_x) + .. + \sum_k m_k \frac{A_k}{\rho_k} W(|r - r_k|, h_z) \tag{21}$$

For calculating the gradient of the quantity we can shift the gradient to each contributing layer.

$$\nabla A(r) = \nabla \left(\begin{array}{c} \sum_j m_j \frac{A_j}{\rho_j} W(|r - r_j|, h_x) + \\ .. + \sum_k m_k \frac{A_k}{\rho_k} W(|r - r_k|, h_z) \end{array} \right) \tag{22}$$

after shifting:

$$\begin{aligned} \nabla A(r) = \sum_j m_j \frac{A_j}{\rho_j} \nabla W(|r - r_j|, h_x) + \\ .. + \sum_k m_k \frac{A_k}{\rho_k} \nabla W(|r - r_k|, h_z) \end{aligned} \tag{23}$$

Similarly Laplace of the quantity:

$$\begin{aligned} \nabla^2 A(r) = \sum_j m_j \frac{A_j}{\rho_j} \nabla^2 W(|r - r_j|, h_x) + \\ .. + \sum_k m_k \frac{A_k}{\rho_k} \nabla^2 W(|r - r_k|, h_z) \end{aligned} \tag{24}$$

Recall that, in Eqn. (13), we need to calculate $(-\nabla p)$, $(\nabla^2 v)$ and external forces to find particle acceleration (a).

Applying Eqn. (23) for $-\nabla p$:

$$\begin{aligned} -\nabla p_i = -\sum_j m_j \frac{p_j}{\rho_j} \nabla W(|r_i - r_j|, h_x) - \\ .. - \sum_k m_k \frac{p_k}{\rho_k} \nabla W(|r_i - r_k|, h_z) \end{aligned} \tag{25}$$

To calculate pressure, Eqn. (15) is used. In order to do this, we need to calculate density using Eqn. (21):

$$\begin{aligned}\rho_i &= \sum_j m_j \frac{\rho_j}{\rho_j} W(|r_i - r_j|, h_x) + .. + \sum_k m_k \frac{\rho_k}{\rho_k} W(|r_i - r_k|, h_z) \\ &= \sum_j m_j W(|r_i - r_j|, h_x) + .. + \sum_k m_k W(|r_i - r_k|, h_z)\end{aligned} \tag{26}$$

As stated previously in Section 3.1, the symmetrization technique presented in [Mül03] does not consider density differences; as a result, it does not produce symmetric forces. But it helps to make forces more proportional, so the computation becomes numerically more stable. A direct analogy in M-SPH should also consider averaging the kernels. But putting kernel averages into account causes too much dampening effect in M-SPH, which results in an unrealistic behavior. The following modification was suitable to our simulation in terms of stability and produced more realistic results in Multiphase SPH:

$$\begin{aligned}-\nabla p(r_i) = &-\sum_j m_j \frac{p_j + p_i}{2\rho_j} \nabla W(|r_i - r_j|, h_x) - \\ &.. - \sum_k m_k \frac{p_k + p_i}{2\rho_k} \nabla W(|r_i - r_k|, h_z)\end{aligned} \tag{27}$$

For similar reasons we find the following modification suitable for Multiphase SPH:

$$\begin{aligned}\mu \nabla^2 v_i = \ &\mu \sum_j m_j \frac{v_j - v_i}{\rho_j} \nabla^2 W(|r_i - r_j|, h_x) + \\ &.. + \mu \sum_k m_k \frac{v_k - v_i}{\rho_k} \nabla^2 W(|r_i - r_k|, h_z)\end{aligned} \tag{28}$$

Similar to classic SPH, acceleration for the particle is computed using Eqn. (13), and then the particle positions are calculated using Verlet integration.

It is important to note that, a simple trick to use lower gravity forces for smaller particles helps to concentrate them near the surfaces which provides much more realistic and detailed surfaces.

4.1 Smoothing Kernels

For density calculations the poly6 kernel introduced by [Mül03] is used:

$$W_{poly6}(r,h) = \frac{315}{64\pi h^9}\begin{cases}(h^2-r^2)^3, & 0 \le r \le h \\ 0, & otherwise\end{cases} \tag{29}$$

This kernel is very simple and provides good stability in simulation. Also in terms of performance, the r term only appears in square form which is useful in implementation as no square root computation will be required in distance calculations.

For pressure calculations the spiky kernel introduced by [Des96] satisfied our needs:

$$W_{spiky}(r,h) = \frac{15}{\pi h^6}\begin{cases}(h-r)^3, & 0 \le r \le h \\ 0, & otherwise\end{cases} \tag{30}$$

For viscosity calculations the viscosity kernel, again, introduced by [Mül03] is used:

$$W_{viscosity}(r,h) = \frac{15}{2\pi h^3}\begin{cases}-\frac{r^3}{2h^3}+\frac{r^2}{h^2}+\frac{h}{2r}-1, & 0 \le r \le h \\ 0, & otherwise\end{cases} \tag{31}$$

4.2 Rendering

In SPH, the simulation results with an implicit definition of the water surface that is being modeled. But a renderable free surface geometry should be generated by polygonization. There are various techniques presented in the literature for polygonizing the surface generated by SPH such as metaballs, marching cubes, point splatting [Mül03], carpet visualization [Kip06].

We have used marching cubes to polygonize the surface generated in our simulation for the sake of rendering quality and implementation availability. Marching cubes is a patented algorithm (patent recently expired [Mar08]) for extracting 3D surface from an isosurface definition.

The basic idea behind marching cubes is as follows: At any point in space we define a voxel, where each corner of the voxel is determined whether it resides in the surface or not. The decision is made by looking at the result of the function at that point. If the result is greater than some threshold value it resides otherwise not. After all corners are evaluated with the test, triangular patches are constructed with this intersection information and these patches are connected to build up the surface.

It is easier to demonstrate the algorithm in its two dimensional version: marching squares.

Consider a grid with the evaluated weight values given in corners:

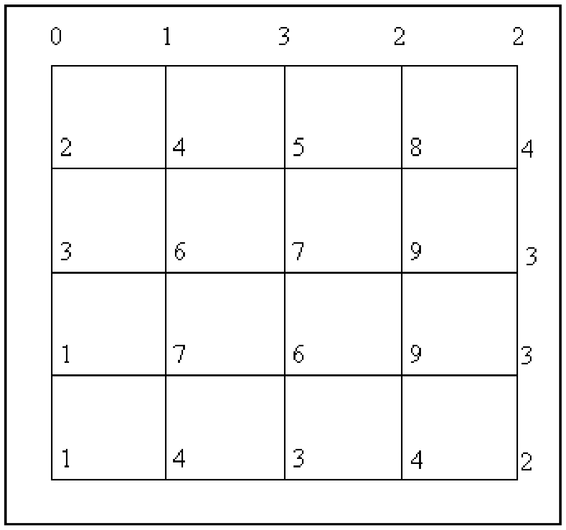


Figure 4.3: Evaluated weights for an implicit surface

Applying threshold value of 5, each corner is selected to be either inside the iso-surface or not:

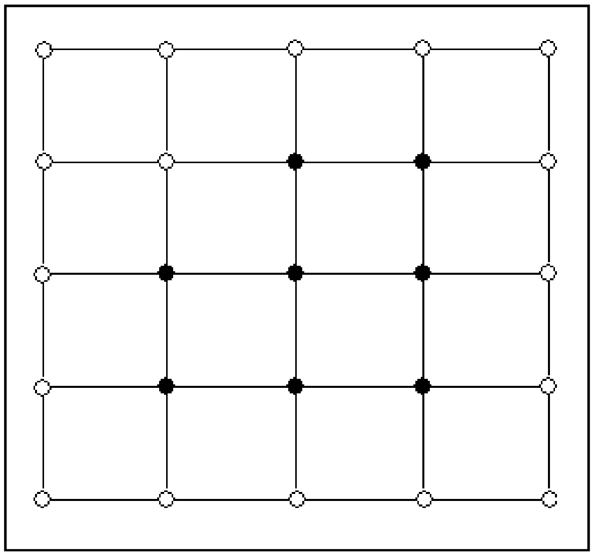


Figure 4.4: Grid-surface intersection data

For each grid cell, we can use the following table to identify the line patch that is contributed to the surface by that cell:

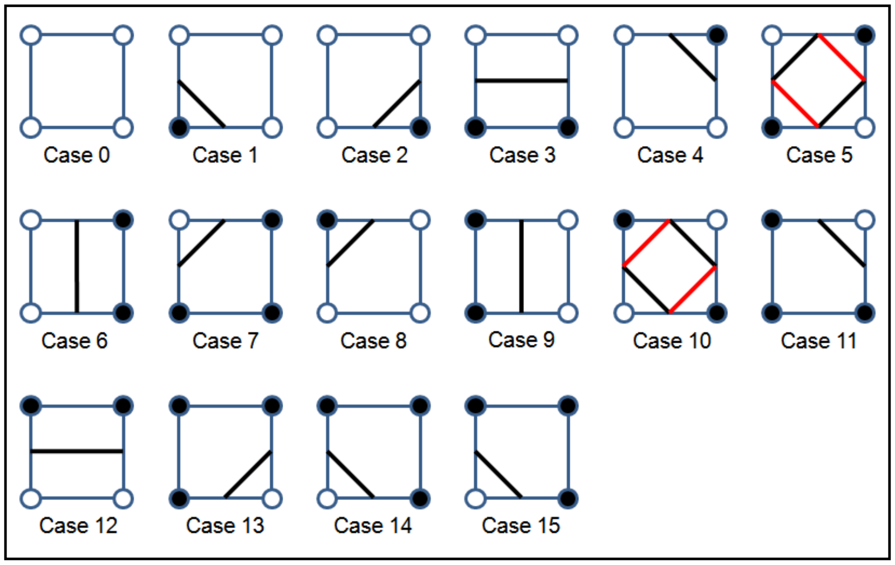


Figure 4.5: Marching squares cases (Bold dots states an intersection with the isosurface)

After applying the table to our grid, we can easily construct the surface:

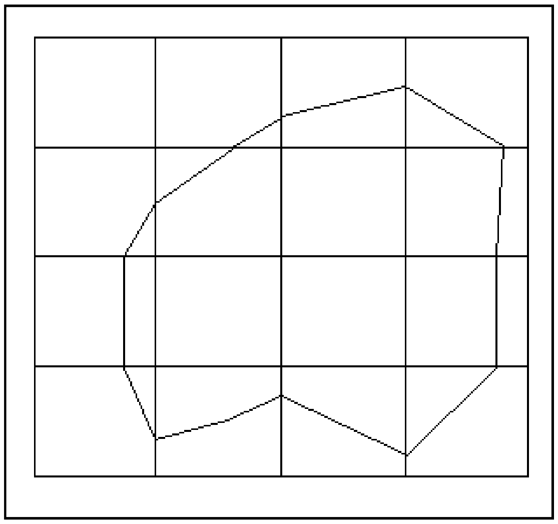
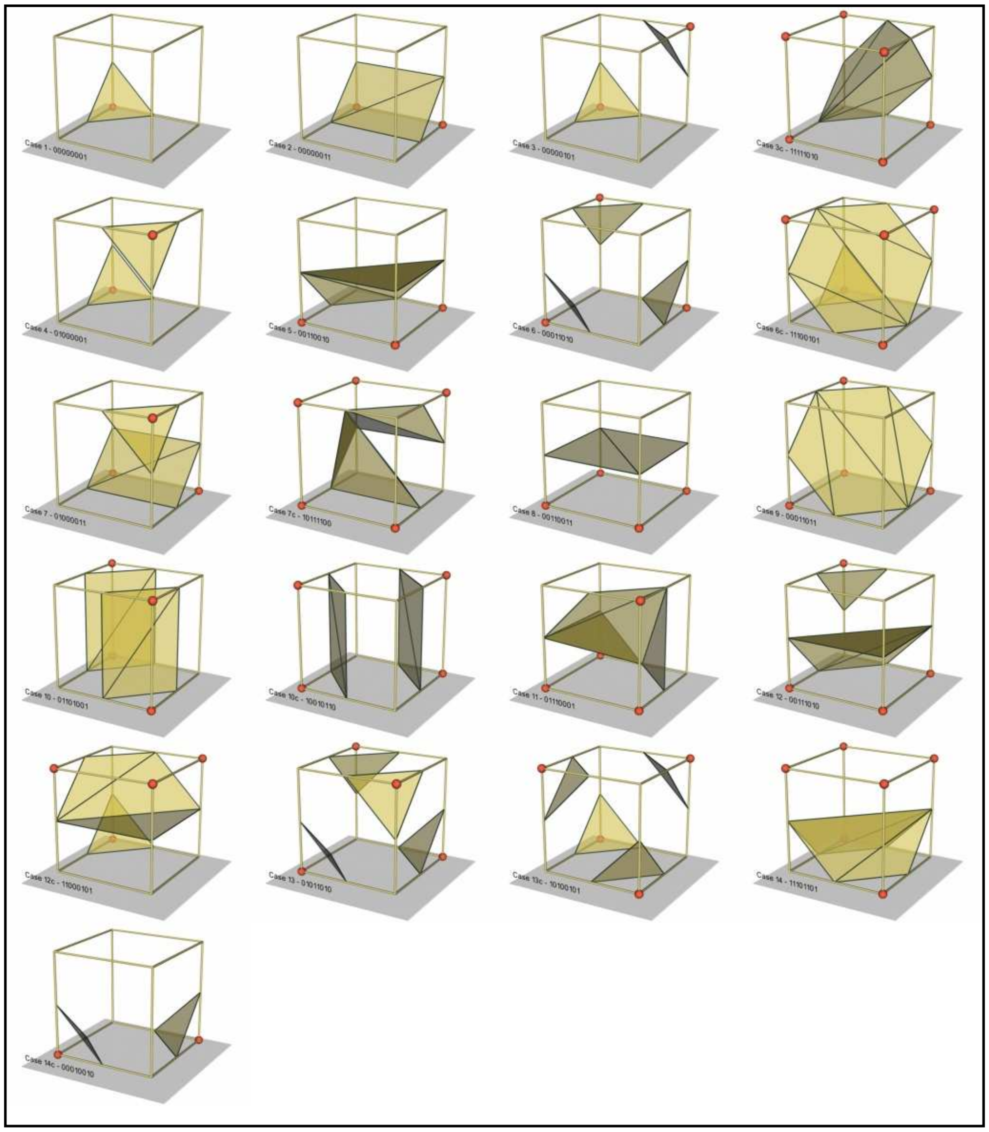


Figure 4.6: Extracted surface

Case 5 and Case 10 are the ambiguous cases for the 2D version of the marching cubes because more than one line patch configuration can be applied for those configurations (red line patches). It is not a problem in the 2D version as it will not break the connectivity of the surfaces. But in 3D these and some other incompatible configurations could cause topology problems (holes in the surfaces) which can be solved with the complementary configurations. Figure 4.7 shows 14 unique configurations of 256 possible cases plus their 7 complementary configurations.



(With permission from Bill Lorensen)

Figure 4.7 Unique and complementary cases of the Marching Cubes Algorithm

To create the surface from Multi-phase SPH model, we have used the density function given in Eqn. (26) as the isosurface function for the marching cubes algorithm. To apply marching cubes, a bounding box is created for the water that is being simulated using particles' coordinates and the maximum smoothing length. Then a grid is created in this bounding box and density at each point of grid is evaluated using Eqn. (26). Finally, the surface is constructed using marching cubes by the volume information created in the previous step.

Chapter 5

Experimental Results

In our experiments, we have used two sizes of water pool and filled them with 3 different configurations of SPH. The simulations performed on a laptop with Intel® Centrino® 1.86 GHz CPU and NVidia® Geforce® Go 6800 graphics card.

First configuration (System 1) is an example of classic SPH approach with 0.2 unit mass for each particle, considered to be first reference system.

The second configuration (System 2) is another classic SPH system with 0.1 unit particle mass. This system is expected to give visually finer detail compared to first configuration with a much lower performance.

The third configuration (System 3) is M-SPH system, our proposed method. We have found that 4 layers are enough for the scope of this experiment. The system is expected to give both visually finer detail and better performance compared to first configuration.

5.1 Performance

The System 1 required 2650 particles to fill the small pool. The system was stable and worked at 15 fps in average.

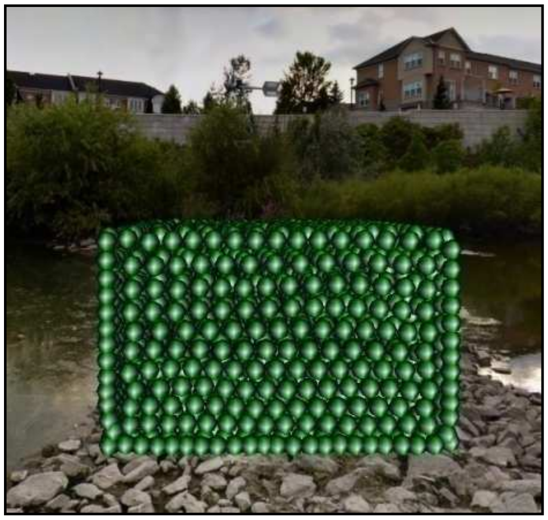


Figure 5.1: Reference system 1 with 2650 particles

The System 2 required 4650 particles while maintaining ~7 fps with an acceptable stability.

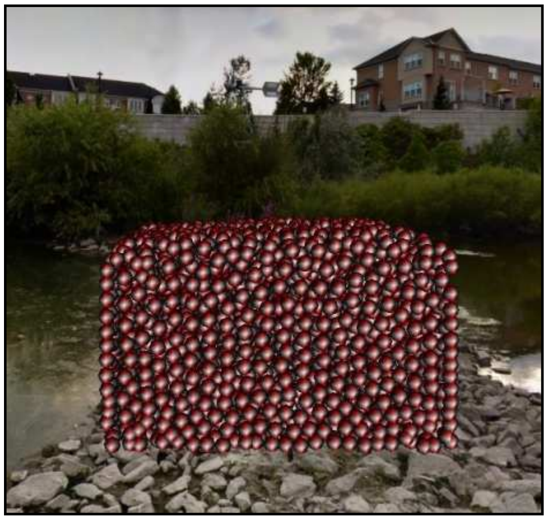


Figure 5.2: Reference system 2 with 4500 particles

The System 3, M-SPH, composed of

800 particles of 0.1 unit mass at 1st layer

400 particles of 0.2 unit mass at 2nd layer

250 particles of 0.4 unit mass at 3rd layer

350 particles of 0.8 unit mass at 4th layer

with total of 1800 particles to fill the small pool. This configuration runs 20 fps with a very good stability.

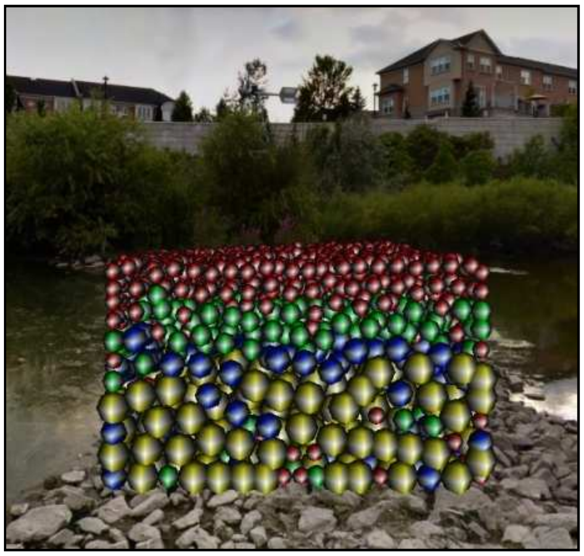


Figure 5.3: The System 3 with 1800 particles

We have expanded the pool to twice its size in order to compare how well these different configurations scale.

We have increased the System 1 particle count by nearly 2 fold to 5100 particles to fill the second pool. The simulation has executed with a nearly acceptable stability at ~6.5 fps:

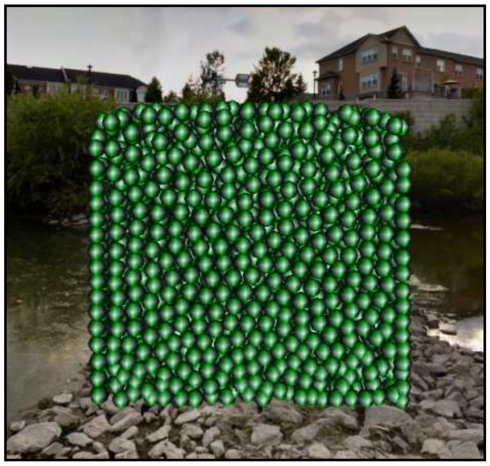


Figure 5.4: Reference system 1 with 5000 particles

Similar to System 1, we have expanded the System 2 particle count to twice its count; nearly 9200 particles. The system has become unstable as the pressure at lower levels of the pool increased dramatically for such small mass. To continue with the experiment, we have tuned smoothing length to a lower value for better stability. Although not reached to an acceptable level, the oscillations have decreased enough to a point that we can capture the results. The performance of the system is ~3.5 fps.

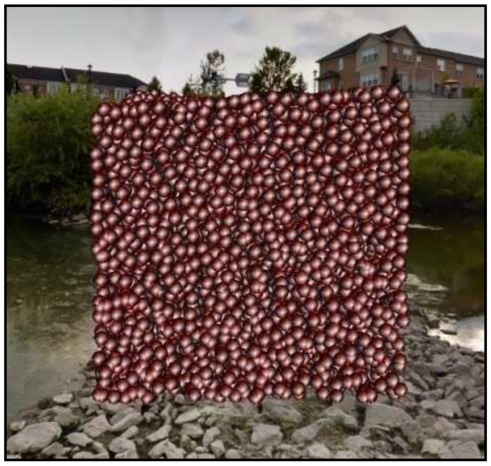


Figure 5.5: Reference system 2 with 9000 particles

We have also increased System 3's particles to 2500 to compensate for the new pool size. The distributions are:

800 particles of 0.1 unit mass at 1st layer,

400 particles of 0.2 unit mass at 2nd layer,

250 particles of 0.4 unit mass at 3rd layer,

1050 particles of 0.8 unit mass at 4th layer.

In the simulation, the system has kept its stability and performed at ~12 fps. As expected, the M-SPH approach has scaled very well while maintaining a good level of stability.

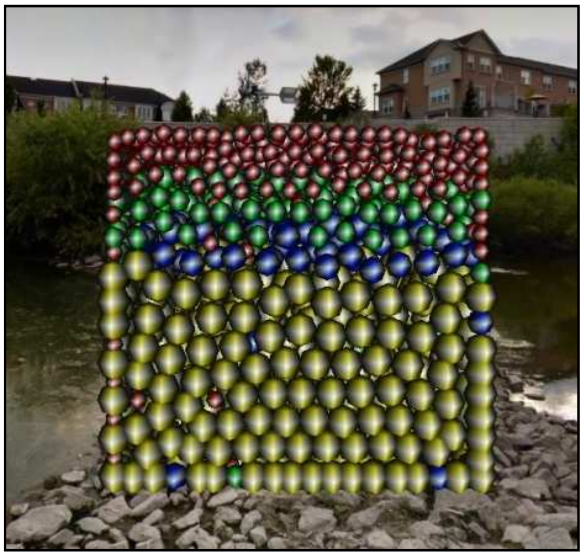


Figure 5.6: The System 3 with 2500 particles

The Table 5.1 summarizes the performance results.

Table 5.1: Summary of the results

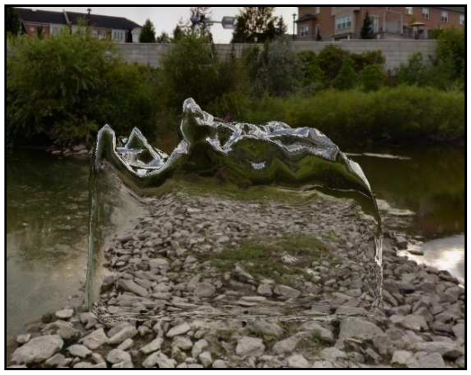
Pool Size	System 1 (SPH) *(Particle Count / FPS)*	System 2 (SPH) *(Particle Count / FPS)*	System 3 (M-SPH) *(Particle Count / FPS)*
x	2650 / 15	4650 / 7	1800 / 20
2x	**5100 / 6.5**	**9200 / 3**	**2500 / 12**
	Results without rendering:		
x	2650 / 22	4650 / 10	1800 / 33.5
2x	5100 / 9	9200 / 4.2	2500 / 24.5

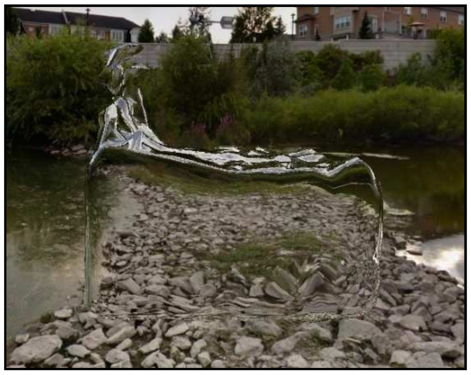
5.2 Visual Results

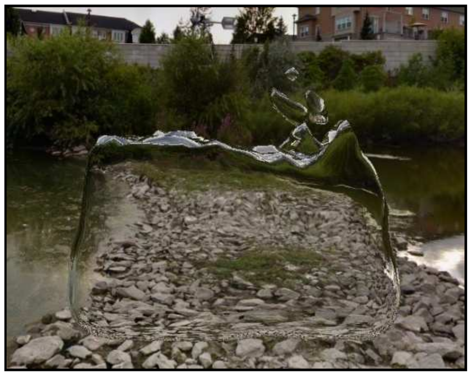
Following are some still frames taken in experiments:

(a)

(b)


(c)


(d)


(e)

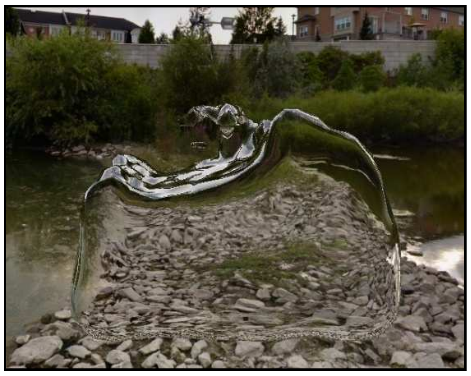

(f)

Figure 5.7: Still frames from the simulation: (a) and (b) for System 1: (c) and (d) for System 2: (e) and (f) for System 3

As can be seen from the results, M-SPH system outperforms the System 1, and provides a greater detail in surfaces. The details are comparable to System 2; additionally the real-time goals are achieved.

Chapter 6

Conclusion

In this research we have introduced a new approach for modeling water behavior with multiple layers of Smoothed Particle Hydrodynamics, called Multiphase SPH. Our approach models different levels of detail in different layers, providing better stability and performance. We were able to achieve nearly 4 times higher frame rates in our experiments with a decreased number of required particles. Considering only the simulation time, frame rate increase reaches up to 6 folds. Also we have shown that our approach scales much better than regular SPH, when the volume increases for the simulated water. The volume increase by 2 fold cause M-SPH frame rates to decrease by %30 while the regular SPH model frame rates decrease by %60.

One possible improvement to our model can be the use of more symmetrically adjusted forces by a better modification of the pressure equations. For example; an equation similar to Eqn. (32) or Eqn. (33) can be utilized [Col06].

$$p_{ij=} \left[\frac{p_i}{{p_i}^2} + \frac{p_j}{{p_j}^2} \right] \tag{32}$$

$$p_{ij=} \left[\frac{p_i + p_j}{2p_i p_j} \right] \tag{33}$$

We have already suggested the use of scaled down gravity forces at layers with smaller particles to keep them near the surfaces at Chapter 4. Although this technique helps to obtain visually more appealing results in our simulations, in real-life scenarios a better approach is needed. One solution is the use of additional attraction forces towards rendered surfaces. To achieve this, one can use an addition term, known as the “color term” in SPH literature, to find the air/water contacts direction, and apply a force in that direction to particles at higher resolution layer, in order to capture detail in surface. A possible interpretation of this technique is applying surface tension in lower detail SPH layers. Also we have not proposed a method to dynamically adapt particle distributions between different layers that could be useful for performing dynamic level of detail.

Most of the CPU time in our simulation time step is spent for executing the marching cubes algorithm. With recent advancement in GPU hardware, the geometry shader capabilities are introduced by chip manufacturers. With help of this new hardware, marching cubes algorithm can be completely executed on the GPU [Tar06]. This can also help to achieve visually more appealing results as the tessellation can be performed in post projection space, additionally with a free level of detail.

If one stays with the CPU surface construction, the carpet visualization method introduced in [Kip06] can also be considered for improving rendering performance. Alternatively, as a potential area for future research, the marching cubes algorithm can be optimized by utilizing SPH's surface tracking capabilities.

Appendix A

Implementation

The simulation system is mostly implemented with C++ using object-oriented techniques. The rendering system uses C code that performs marching cubes surface construction and CG code performing the shading with reflection and refraction, both previously available on internet [Ama08]. Microsoft Windows XP is selected as the target platform, and Microsoft Visual Studio 2003 integrated development environment is used for the development activities.

To keep real-time boundaries, the animation system pre-allocates the memory for the particle system using a particle pool. Also the grid and surface points used in the Marching cubes are allocated at the initialization by the simulation system. These pre-allocations all together prevent the dynamic memory allocations from the heap that drastically increases the performance. Additionally, by the help of the particle pool, which keeps particles sequentially, the cache hit ratio increases in the calculations.

A. 1 Implementation Details

Figures A.1.1 to A.1.4 show the major classes and their relationships.

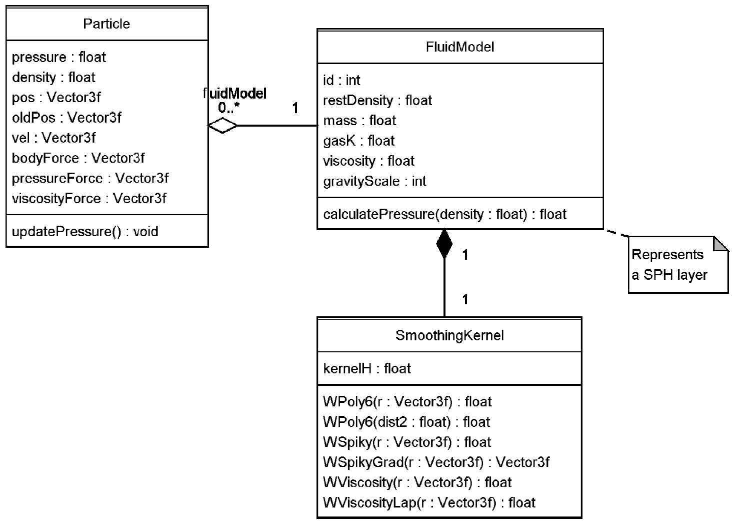


Figure A.1.1: The Particle model

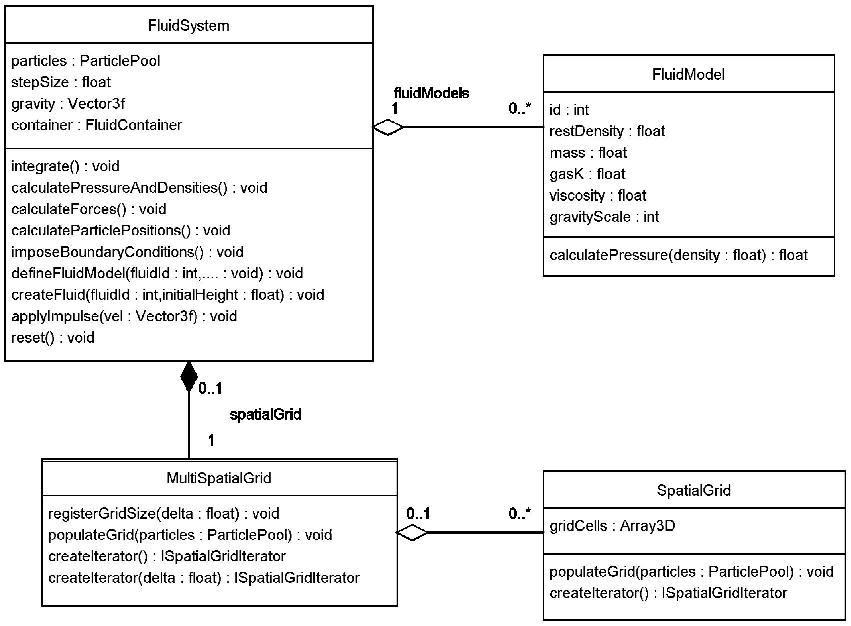

FluidSystem
particles : ParticlePool
stepSize : float
gravity : Vector3f
container : FluidContainer
integrate() : void
calculatePressureAndDensities() : void
calculateForces() : void
calculateParticlePositions() : void
imposeBoundaryConditions() : void
defineFluidModel(fluidId : int,.... : void) : void
createFluid(fluidId : int,initialHeight : float) : void
applyImpulse(vel : Vector3f) : void
reset() : void
fluidModels
1
0..*
FluidModel
id : int
restDensity : float
mass : float
gasK : float
viscosity : float
gravityScale : int
calculatePressure(density : float) : float
0..1
spatialGrid
1
MultiSpatialGrid
registerGridSize(delta : float) : void
populateGrid(particles : ParticlePool) : void
createIterator() : ISpatialGridIterator
createIterator(delta : float) : ISpatialGridIterator
0..1
0..*
SpatialGrid
gridCells : Array3D
populateGrid(particles : ParticlePool) : void
createIterator() : ISpatialGridIterator

Figure A.1.2: The Simulation subsystem

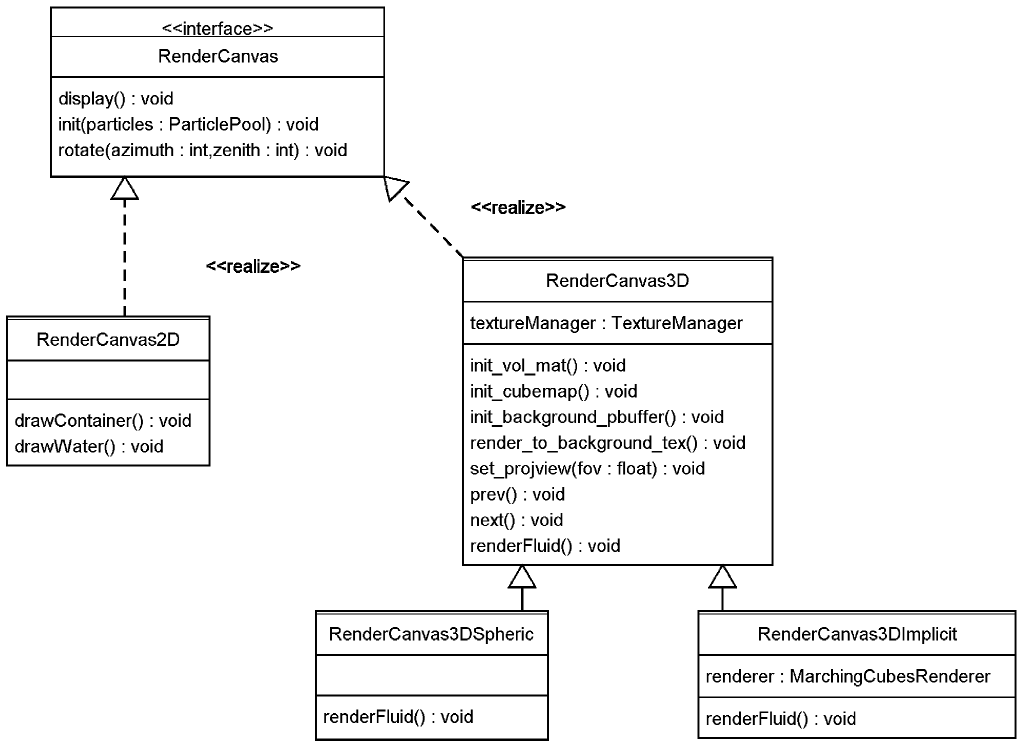


Figure A.1.3: The Rendering subsystem

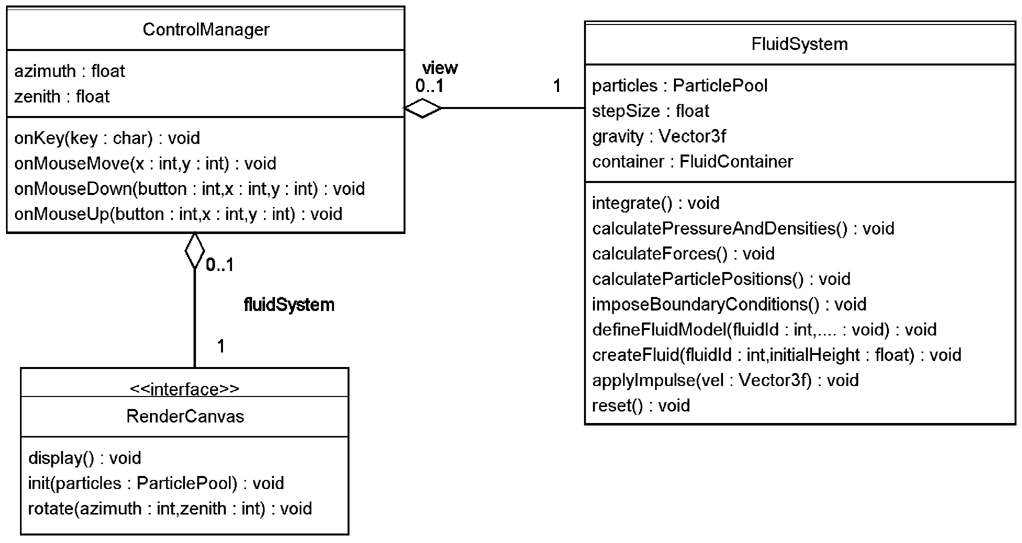

ControlManager
azimuth : float
zenith : float
onKey(key : char) : void
onMouseMove(x : int,y : int) : void
onMouseDown(button : int,x : int,y : int) : void
onMouseUp(button : int,x : int,y : int) : void
view
0..1
1
FluidSystem
particles : ParticlePool
stepSize : float
gravity : Vector3f
container : FluidContainer
integrate() : void
calculatePressureAndDensities() : void
calculateForces() : void
calculateParticlePositions() : void
imposeBoundaryConditions() : void
defineFluidModel(fluidId : int,.... : void) : void
createFluid(fluidId : int,initialHeight : float) : void
applyImpulse(vel : Vector3f) : void
reset() : void
0..1
fluidSystem
1
<<interface>>
RenderCanvas
display() : void
init(particles : ParticlePool) : void
rotate(azimuth : int,zenith : int) : void

Figure A.1.4: The Control subsystem

The general simulation step is as following:

I. populate the grid
II. calculate pressure and densities
III. calculate forces
IV. calculate particle positions
V. impose boundary conditions
VI. render

The details for individual steps are given in A.2.

The general SPH mechanism requires a heavy use of spatial queries (i.e., finding particles inside a specific radius of a point). For each layer this radius is equal to the smoothing length. For this reason we have used different grids for each layer with a resolution equal to the corresponding layer's smoothing length.

The population of the grids is a very fast operation that consists of the calculation of the index for the particle and then, appending the particle to the end of the list at that index:

```
for each particle
    i,j,k = (particle->pos - gridPosition) / smoothingLength;
    mGridEntries[i,j,k]->append(particle);
```

Accessing to the particles in smoothing length radius of a position is similar:

```
getParticlesAt(Position pos){
    i,j,k = (pos - gridPosition) / smoothingLength;
    return mGridEntries[i,j,k];
}
```

The source will be available under BSD license on the internet, freely usable and downloadable from a website.

A. 2 Detailed Pseudo-codes for Simulation Step

```
//Calculates the particle densities and pressures
calculatePressureAndDensities(){
  for each particle
    particle->denisity = 0

  for each particle
    FluidModel fluidModel = particle->fluidModel;
    for each neighbour
      r = particle->pos - neighbour->pos;
      dens = fluidModel->particleMass * fluidModel->kernel->WPoly6(r);
      neigh->density += dens;

  for each particle
    particle->updatePressure();
}


//Calculates total force acting on particles
calculateForces(){
  for each particle
    particle->bodyForce = gGravity * particle->density;
    particle->pressureForce = 0;
    particle->viscosityForce = 0;

  for each particle
    FluidModel fluidModel = particle->fluidModel;
    for each neighbour
      r = particle->pos - neighbour->pos;
      if r.length >= fluidModel
        skip neighbour;

      //Pressure
      force = fluidModel->particleMass * (particle->pressure + neigh->pressure)
                / (2 * particle->density) * fluidModel->kernel.WSpikyGrad(r);
      neigh->pressureForce += force;

      //Viscosity
      force = particle->fluidModel->mass * (particle->vel - neighbour->vel)
                * (fluidModel->viscosity + neigh->fluidModel->viscosity)
                / (2 * particle->density)* fluidModel.kernel.WViscosityLap(r);
      neigh->viscosityForce += force;
}
```

```
//Calculates particle positions by Verlet integration
calculateParticlePositions(){
  for each particle
    tmp = particle->pos;
    force = particle->bodyForce + particle->pressureForce
               + particle->viscosityForce;
    //Verlet integration
    newPos = 2 * particle->pos - particle->oldPos
               + force / particle->density * stepSize^2;
    particle->oldPos = tmp;
}

//Imposes the boundary conditions, i.e., container limitation
imposeBoundaryConditions(){
  for each particle
    if passesThroughMesh(particle->oldPos , particle->pos)
      particle->pos = reflectFromMesh(particle->oldPos , particle->pos);

    //Update particle velocity
    particle->vel = (particle->pos - particle->oldPos) / stepSize;
}
```

Bibliography

[Enr02] D. Enright, S. Marschner, and R. Fedkiw, "Animation and rendering of complex water surfaces," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 736-744, 2002.

[Fos01] N. Foster and R. Fedkiw, "Practical animation of liquids," in *SIGGRAPH '01: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, New York, NY, 2001, pp. 23-30.

[Jen01] H. W. Jensen, *Realistic image synthesis using photon mapping*. Natick, MA: A. K. Peters, Ltd., 2001.

[Sta95] J. Stam and E. Fiume, "Depicting fire and other gaseous phenomena using diffusion processes," in *SIGGRAPH '95: Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, New York, NY, 1995, pp. 129-136.

[Des96] M. Desbrun and M.-P. Gascuel, "Smoothed particles: a new paradigm for animating highly deformable bodies," in *Proceedings of the Eurographics workshop on Computer Animation and Simulation '96*, Poitiers, France, 1996, pp. 61-76.

[Cla05] S. Clavet, P. Beaudoin, and P. Poulin, "Particle-based viscoelastic fluid simulation," in *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Los Angeles, California, 2005, pp. 219-228.

[Mül05] M. Müller, B. Solenthaler, R. Keiser, and M. Gross, "Particle-based fluid-fluid interaction," in *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Los Angeles, California, 2005, pp. 237-244.

[Hon08] W. Hong, D. H. House, and J. Keyser, "Adaptive particles for incompressible fluid simulation," *The Visual Computer*, vol. 24, no. 7-9, pp. 535-543, Jul. 2008.

[Luc77] L. B. Lucy, "A numerical approach to the testing of the fission hypothesis," *The Astronomical Journal*, no. 82, pp. 1013-1024, 1977.

[Gin77] R. Gingold and J. Monaghan, "Smoothed particle hydrodynamics - theory and application to non-spherical stars," *Monthly Notices of the Royal Astronomical Society*, no. 375, 1977.

[GRL03] G. R. Liu and M. B. Liu, "Smoothed particle hydrodynamics (SPH)," in *Smoothed Particle Hydrodynamics: A Meshfree Particle Method*. World Scientific Publishing Company, 2003.

[Bri06] R. Bridson, R. Fedkiw, and M. Muller-Fischer, "Fluid simulation: SIGGRAPH 2006 Course Notes," in *SIGGRAPH '06: ACM SIGGRAPH 2006 Courses*, Boston, Massachusetts, 2006, pp. 1-87.

[Kip06] P. Kipfer and R. Westermann, "Realistic and interactive simulation of rivers," in *GI '06: Proceedings of Graphics Interface 2006*, Quebec, Canada, 2006, pp. 41-48.

[Mar08] Wikipedia: The Free Encyclopedia. [Online]. http://en.wikipedia.org/wiki/Marching_cubes

[Col06] F. Colin, R. Egli, and F. Y. Lin, "Computing a null divergence velocity field using smoothed particle hydrodynamics," *J. Comput. Phys.*, vol. 217, no. 2, pp. 680-692, 2006.

[Tar06] S. Tariq, "Next Generation Effects," in *Siggraph Exhibitor Sessions*, 2006.

[Ama08] T. Amada. Real-Time Particle-Based Fluid Simulation. [Online]. http://www.ss.iij4u.or.jp/~amada/fluid/

[Mül03] M. Müller, D. Charypar, and M. Gross, "Particle-based fluid simulation for interactive applications," in *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, San Diego, California, 2003, pp. 154-159.

[Den08] Wikipedia: The Free Encyclopedia. [Online]. http://en.wikipedia.org/wiki/Density

[Vis08] Wikipedia: The Free Encyclopedia. [Online]. http://en.wikipedia.org/wiki/Viscosity

[Wat08] Wikipedia: The Free Encyclopedia. [Online]. http://en.wikipedia.org/wiki/Water_(molecule)#Compressibility

[Ref08] Wikipedia: The Free Encyclopedia. [Online]. http://en.wikipedia.org/wiki/Refraction

[Flu08] Wikipedia: The Free Encyclopedia. [Online]. http://en.wikipedia.org/wiki/Fluid_dynamics

[Der08] Wikipedia: The Free Encyclopedia. [Online]. http://en.wikipedia.org/wiki/Derivation_of_the_Navier–Stokes_equations#Newtonian_fluid