Physically-Based Fluid Simulation for Computer Graphics

BY
Brian Trevethan
Bachelor of Science, Computer Science
Colorado State University
May, 2004

Submitted in partial fulfillment of the requirements
for the degree of Master of Science
in the graduate studies program
of DigiPen Institute Of Technology
Redmond, Washington
United States of America

Fall
2007

Thesis Advisor: Dr. Xin Li

DIGIPEN INSTITUTE OF TECHNOLOGY

GRADUATE STUDY PROGRAM

DEFENSE OF THESIS

THE UNDERSIGNED VERIFY THAT THE FINAL ORAL DEFENSE OF THE MASTER OF SCIENCE THESIS OF Brian Trevethan

HAS BEEN SUCCESSFULLY COMPLETED ON ______

TITLE OF THESIS: Physically-Based Fluid Simulation for Computer Graphics

MAJOR FIELD OF STUDY: COMPUTER SCIENCE

COMMITTEE:

Dr. Xin Li, Chair	Dr. Martin Weinless
Charles Duba	

APPROVED:

Matt Klassen	Date		Date
Graduate Program Director		Associate Dean	
Dr. Xin Li	Date		Date
Department of Computer Science		Dean	

INSTITUTE OF DIGIPEN INSTITUTE OF TECHNOLOGY

PROGRAM OF MASTER'S DEGREE

THESIS APPROVAL

DATE: ______________________________

BASED ON THE CANDIDATE'S SUCCESSFUL ORAL DEFENSE, IT IS RECOMMENDED THAT THE THESIS PREPARED BY

Brian Trevethan

ENTITLED

Physically-Based Fluid Simulation for Computer Graphics

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF COMPUTER SCIENCE FROM THE PROGRAM OF MASTER'S DEGREE AT DIGIPEN INSTITUTE OF TECHNOLOGY.

____

Dr. Xin Li
Thesis Advisory Committee Chair

____

Matt Klassen
Director of Graduate Study Program

____

Associate Dean

____

Dean of Faculty

TABLE OF CONTENTS

Chapter 1
INTRODUCTION

Fluids are present in many parts of our daily lives. From the milk in our cereal bowls to the wind blowing in our face when we ride our bikes, fluids are everywhere. It should come as no surprise then that there has been a great deal of interest in using computers to model this complex and interesting phenomenon. Luckily, engineers in the field of Computational Fluid Dynamics (CFD) have devised extremely accurate methods to model fluid motion. The drawback is that many of these methods are not computationally efficient enough for a computer graphics application where the major concern is to quickly produce fluid motion that appears to be realistic as opposed to motion that is truly realistic.

This paper presents a number of techniques that have been developed by the computer graphics community to physically simulate the motion of fluids. Although the methods are generally applicable to all fluids, particular attention will be directed toward simulating the motion of water, as that is the primary concern of this researcher.

Chapter 2

VECTOR CALCULUS REVIEW

This section is intended for readers that may not deal with vector calculus on a day-to-day basis. It will introduce basic concepts and principles. The discussion is intended for those with a mathematical background that have either never had a reason to deal with vector calculus or those that haven't had to use their knowledge of this field for a long time. It is not intended to be a comprehensive review, but should give enough to help understand the basic concepts that are going to be discussed in the following sections.

2.1 Scalar Fields and Vector Fields

A scalar field is defined as a scalar valued function of position. For example, $t(x, y, z)$ might be used to represent the temperature in a room. Temperature is a scalar value and it is defined at every point in the room, so temperature is a scalar field. In another example, $p = P(\mathbf{r})$ where $\mathbf{r} = (x, y, z)$ might be used to represent the pressure in a fluid.

A vector field is very similar to a scalar field, except that it is a vector-valued function of position. In this paper $\mathbf{u} =< u, v, w >$ where u, v and w are all functions of x, y and z such that $u = u(x, y, z)$, $v = v(x, y, z)$ and $w = w(x, y, z)$. $\mathbf{u}$ will commonly be used to represent the velocity of a fluid at any given point.

2.2 Gradient

The gradient of scalar field s is denoted as, ∇s where the symbol ∇ is known as "nabla" or "del". The gradient of a scalar field is defined as

$$\nabla s = \left\{ \frac{\partial s}{\partial x}, \frac{\partial s}{\partial y}, \frac{\partial s}{\partial z} \right\}$$

. In English, the gradient of a scalar field is the vector of partial derivatives with respect to each of the coordinate axes. From a physical perspective, the gradient represents the direction in which the scalar field is increasing. The magnitude of the gradient vector is equal to the rate of change. As a concrete example, let $s(\mathbf{r}) = xy + z^2 + yz^3$. Then the gradient of s, ∇s, is:

$$\nabla s = (y, x + z^3, 2z + 3yz^2) \tag{2.2.1}$$

The gradient operator can also be applied to a vector field. The gradient of an n-dimensional vector field is an nxn matrix of partial derivatives. The gradient of vector field, $\mathbf{u} =< u, v, w >$, is

$$\begin{pmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} & \frac{\partial u}{\partial z} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} & \frac{\partial v}{\partial z} \\ \frac{\partial w}{\partial x} & \frac{\partial w}{\partial y} & \frac{\partial w}{\partial z} \end{pmatrix} = \nabla \mathbf{u}$$

2.3 Flux

Flux is defined as the total volume of fluid passing through a surface per unit time [22]. Given a surface, S, that has a normal, $\mathbf{n}$, located inside of a vector field $\mathbf{u}$ let δS represent a small subsurface of S. Let the area of δS be defined as $\delta x \delta y = A$. First assume that $\mathbf{u}$ is parallel to $\mathbf{n}$ and $|\mathbf{u}| = c$ where c is a constant. In time t a "block" of fluid of size ctA moves through S. Therefore, the flux, Q, is $Q = cA$.

Now suppose that $\mathbf{u}$ is not parallel to $\mathbf{n}$ and $|\mathbf{u}|$ is no longer constant. See figure 2.1. Since

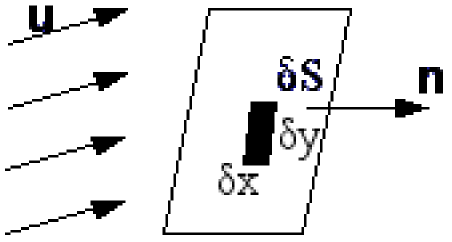


Figure 2.1: $\mathbf{u}$ represents the vector field. The shaded region, δS represents a sub-section of the larger surface S. $\mathbf{n}$ reperesents the outward pointing normal to S.

$\mathbf{u}$ is no longer parallel to $\mathbf{n}$ then the only portion of $\mathbf{u}$ contributing to the flux is the portion that is in the direction of $\mathbf{n}$. This is simply $\mathbf{u} \cdot \mathbf{n}$. Summing the contribution of the flux of each small δS over the entire surface S yields the definition of flux

$$Q = \oiint_S \mathbf{u} \cdot \mathbf{n} \delta x \delta y \tag{2.3.1}$$

2.4 Divergence

The concept of divergence is closely related to that of the gradient. While the gradient acts on a scalar field and results in a vector field, the divergence acts on a vector field and results in a scalar field. The divergence of vector field $\mathbf{v} = (v_1, v_2, v_3)$ is denoted as $\nabla \cdot \mathbf{v}$ and is defined as:

$$\nabla \cdot \mathbf{v} = \frac{\partial v_1}{\partial x} + \frac{\partial v_2}{\partial y} + \frac{\partial v_3}{\partial z} = \left\{ \frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z} \right\} \cdot (v_1, v_2, v_3) \tag{2.4.1}$$

. From the last term in equation (2.4.1) it is shown that the divergence of a vector field can be thought of as the dot product of the "nabla" operator and the vector field.

As a simple example, let the vector field $\mathbf{v}$ be defined as $\mathbf{v} = (x^2+2z, y^2z^3-6x+4, x^2yz^4-2x)$. The divergence of $\mathbf{v}$ is

$$\nabla \cdot \mathbf{v} = 2x + 2yz^3 + 4x^2yz^3 \tag{2.4.2}$$

The divergence of $\mathbf{u}$ is equal to the flux through the surface of a small volume, δV, surrounding

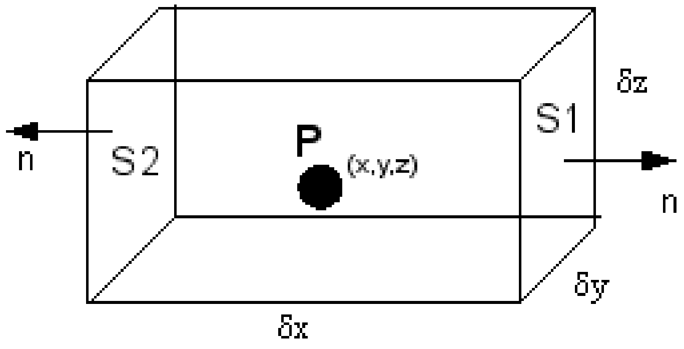


Figure 2.2: A sub-volume, δV_i.

any point in the field divided by the volume of δV. Mathematically this can expressed as

$$\nabla \cdot \mathbf{u} = \lim_{\delta V \to 0} \frac{1}{\delta V} \oiint_{\delta S} \mathbf{u} \cdot \mathbf{n} \delta S \tag{2.4.3}$$

where δV is a small volume surrounding a point P with surface δS and $\mathbf{n}$ is the outward pointing normal of δS. Assume that δV is a small rectangular box with sides δx, δy and δz. See figure 2.2. Since the box has six faces then the integral around the entire box is the sum of the six face integrals. Starting with the surface labeled S_1 in figure 2.2, $\mathbf{n}_1 = (1, 0, 0)$, $\delta S = \delta y * \delta z$. Let $\mathbf{u} = (u, v, w)$.

Therefore, $\mathbf{u} \cdot n = u$. The point at the center of the surface is $c_1 = (P_x + \delta x/2, y, z)$. Since the area of surface S_1 is assumed to be very small then the surface integral can be given by

$$\oiint_{S_1} \mathbf{u} \cdot \mathbf{n} \delta x \delta y \approx u_1(P_x + \delta x/2, y, z)\ . \tag{2.4.4}$$

A similar argument can be made for the surface S_2.

$$\oiint_{S_2} \mathbf{u} \cdot \mathbf{n} \delta x \delta y \approx -u_1(P_x - \delta x/2, y, z)\ . \tag{2.4.5}$$

Adding both integrals together and reducing gives

$$\oiint_{S_1+S_2} \mathbf{u} \cdot \mathbf{n} \delta x \delta y \approx \frac{\partial u}{\partial x} \delta V \tag{2.4.6}$$

Since divergence is defined as $\delta V \to 0$ then equation (2.4.6) is the true divergence with respect to the surface whose normals are parallel to the x-axis. The same procedure can be repeated for y and z.

2.5 Divergence Theorem

The divergence theorem states that the total amount of expansion of $\mathbf{u}$ within the volume V is equal to the flux out of the surface S. Mathematically this can be expressed as

$$\iiint_V \nabla \cdot \mathbf{u} dV = \oiint_S \mathbf{u} \cdot \mathbf{n} dS \tag{2.5.1}$$

To prove this theorem assume that volume V is divided into a bunch of very small sub-volumes, δV_i each with surface δS_i. In each δV_i the divergence is

$$\nabla \cdot \mathbf{u} = \lim_{\delta V \to 0} \frac{1}{\delta V} \oiint_{\delta S} \mathbf{u} \cdot \mathbf{n} \delta S \tag{2.5.2}$$

Multiplying both sides by δV_i and summing for δV_i yields

$$\sum_i \nabla \cdot \mathbf{u} \delta V \approx \sum_i \oiint \mathbf{u} \cdot \mathbf{n} dS \tag{2.5.3}$$

Taking the limit as $\delta V_i \to 0$ makes the left-hand side equal to the volume integral, $\iiint_V \nabla \cdot u dV$. For the right-hand side, consider two adjacent sub-volumes as shown in figure 2.3. Note that

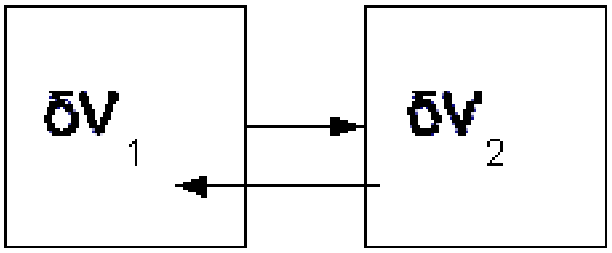


Figure 2.3: A 2D representation of two sub-volumes

$\mathbf{u} \cdot \mathbf{n_1} + \mathbf{u} \cdot \mathbf{n_2} = 0$. This is because the normals are equal and opposite. Thus all interior surfaces will cancel each other leaving only the exterior edges. Therefore,

$$\iiint_V \nabla \cdot \mathbf{u} dV = \oiint_S \mathbf{u} \cdot \mathbf{n} dS \tag{2.5.4}$$

Chapter 3

EQUATIONS OF MOTION - THE NAVIER STOKES EQUATIONS

In this section the Navier-Stokes equations that are used to describe the conservation of momentum and mass for a fluid are presented. The presentation is followed by a derivation of each equation.

In the 19th century, two mathematicians, Claude-Louis Navier and George Gabriel Stokes developed the Navier-Stokes equations that describe the motion of a fluid [29]. For incompressible flows there are two equations

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla)\mathbf{u} - \frac{1}{\rho}(\nabla p) + \nu \nabla^2 \mathbf{u} + \mathbf{g} \tag{3.0.1}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{3.0.2}$$

where $\mathbf{u} = (u, v, w)$ represents the velocity of the flow, ρ is the density of the fluid, p is the pressure, ν is the kinematic viscosity and $\mathbf{g}$ is the acceleration due to any external forces. Equation (3.0.2) indicates a conservation of mass and equation (3.0.1) describes how the rate of change of the velocity of a flow is related to the process of advection, the acceleration due to pressure, the acceleration due to viscosity and acceleration resulting from external forces such as gravity. It is important for the reader to understand that equation (3.0.1) is a vector-valued equation that actually represents three equations. Equation (3.0.2) is sometimes referred to as the continuity equation.

3.1 Conservation of Mass Equation

Consider a fluid with density given by the scalar field $\rho(\mathbf{x}, t)$ and velocity given by the vector field $\mathbf{u}(\mathbf{x}, t)$. Let V be an arbitrary stationary volume with surface S and outward pointing normal $\mathbf{n}$. The total mass of the fluid inside of V is defnied as

$$M = \iiint_V \rho dV \tag{3.1.1}$$

The rate at which mass enters the volume, r, is defined as the flux through the surface and can be expressed as

$$r = -\oiint \rho\mathbf{u}\cdot\mathbf{n}dS \tag{3.1.2}$$

The minus sign is used because $\mathbf{n}$ is assumed to point outward. The rate of change of mass in the volume must be equal to the rate of mass flow into or out of V_i. Thus, we get

$$\frac{d}{dt}\iiint_V \rho dV = -\oiint \rho\mathbf{u}\cdot\mathbf{n}dS \tag{3.1.3}$$

The surface integral on the right-hand side can be written as a volume integral using the divergence theorem, also the order of the derivative and the integral on the left-hand side can be interchanged thus yielding

$$\iiint_V \frac{\partial\rho}{\partial t}dV = -\iiint_V \nabla\cdot(\rho\mathbf{u})dV \tag{3.1.4}$$

Since the integrals are over the same volume then we can add the right-hand side to the left-hand side and place both integrands under the same integral

$$\iiint \frac{\partial\rho}{\partial t} + \nabla\cdot(\rho\mathbf{u})dV = 0 \tag{3.1.5}$$

If the chosen volume, V, is small enough then the value of the integrand can be considered constant throughout the volume so the integral can be dropped and we're left with

$$\frac{\partial\rho}{\partial t} + \nabla\cdot(\rho\mathbf{u}) = 0 \tag{3.1.6}$$

Since the density is defined to be constant, then it can moved outside of the differentiation, giving

$$\frac{\partial\rho}{\partial t} + \nabla\cdot(\rho\mathbf{u}) = \frac{\partial\rho}{\partial t} + \rho\nabla\cdot(\mathbf{u}) = 0 \tag{3.1.7}$$

Again using the fact that density is constant with respect to both time and space then the above equation reduces further giving

$$0 + \rho\nabla\cdot(\mathbf{u}) = 0 \tag{3.1.8}$$

Therefore, $\nabla \cdot \mathbf{u} = 0$ must be true.

3.2 Conservation of Momentum Equation

The conservation of momentum says that the rate of change of momentum through a fixed volume must equal the flux through the surface plus any external forces, $\mathbf{F}$. Assume a rectangular volume, dV of dimensions δx, δy and δz. The rate of change of momentum can be defined as

$$\frac{\partial \rho \mathbf{u}}{\partial t} dV \tag{3.2.1}$$

where $\mathbf{u} = (u, v, w)$. The flux of momentum in the x-direction through the face whose normal is $(-1, 0, 0)$ is

$$\rho(\mathbf{u} \cdot \mathbf{n})Au = -\rho u * u * A = -\rho u * u * \delta y \delta z \tag{3.2.2}$$

where A is the area of the surface. The flux through the face whose normal is $(1, 0, 0)$ is

$$\left(\rho u * u + \frac{\partial \rho u * u}{\partial y} \delta x\right) \delta y \delta z \tag{3.2.3}$$

The flux in the x-direction of the face whose normal is $(0, -1, 0)$ is

$$-\rho v * u * \delta x \delta z \ . \tag{3.2.4}$$

The flux through the opposite face whose normal is $(0, 1, 0)$ is

$$\left(\rho v * u + \frac{\partial \rho v * u}{\partial x} \delta y\right) \delta x \delta z \tag{3.2.5}$$

The flux in the x-direction of the face whose normal is $(0, 0, -1)$ is

$$-\rho w * u * \delta x \delta z \ . \tag{3.2.6}$$

The flux through the opposite face whose normal is $(0, 0, 1)$ is

$$\left(\rho w * u + \frac{\partial \rho w * u}{\partial x} \delta y\right) \delta x \delta z \tag{3.2.7}$$

Summing all six faces gives

$$\begin{aligned}
Q_x &= -\rho u * u * \delta y \delta z + \left(\rho u * u + \frac{\partial \rho u * u}{\partial x}\delta x\right)\delta y \delta z && (3.2.8)\\
&\quad -\rho v * u * \delta x \delta z + \left(\rho v * u + \frac{\partial \rho v * u}{\partial y}\delta y\right)\delta x \delta z && (3.2.9)\\
&\quad -\rho w * u * \delta x \delta z + \left(\rho w * u + \frac{\partial \rho w * u}{\partial z}\delta y\right)\delta x \delta z && (3.2.10)\\
&= \frac{\partial \rho u * u}{\partial x}\delta x \delta y \delta z + \frac{\partial \rho v * u}{\partial y}\delta y \delta x \delta z + \frac{\partial \rho w * u}{\partial z}\delta y \delta x \delta z && (3.2.11)\\
& && (3.2.12)
\end{aligned}$$

Including the time derivative yields the final equation for the total flux in the x-direction as

$$\sum F_x = \left(\frac{\partial \rho u}{\partial t} + \frac{\partial \rho u * u}{\partial x} + \frac{\partial \rho v * u}{\partial y} + \frac{\partial w * u}{\partial z}\right)\delta x \delta y \delta z \tag{3.2.13}$$

Simplifying and extending the same concept for y and z yields

$$\sum \mathbf{F}_x = \rho\left(\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z}\right)\delta X \delta Y \delta Z \tag{3.2.14}$$

$$\sum \mathbf{F}_y = \rho\left(\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + w\frac{\partial v}{\partial z}\right)\delta X \delta Y \delta Z \tag{3.2.15}$$

$$\sum \mathbf{F}_z = \rho\left(\frac{\partial w}{\partial t} + u\frac{\partial w}{\partial x} + v\frac{\partial w}{\partial y} + w\frac{\partial w}{\partial z}\right)\delta X \delta Y \delta Z \tag{3.2.16}$$

The rate of change of momentum is equal to the total force acting on the fluid. Fluids are subject to two different forces. These forces are surface forces and body forces such as gravity. Body forces will not be discussed here, but their inclusion into the system are trivial.

Surface forces occur due to pressure and viscosity. They can be described by $\sigma_{i,j}$ where subscript i indicates the normal direction of the face on which the force is acting and j indicates the direction of the stress. σ is called the stress tensor. The force due to stress is defined as the quantity of the stress tensor times the area of the face. For faces with normals in the x-direction, then the forces in the x-direction are

$$-\sigma_{xx}\delta y \delta z \quad \text{and} \quad \left(\sigma_{xx} + \frac{\partial \sigma_{xx}}{\partial x}\delta x\right)\delta y \delta z \tag{3.2.17}$$

Their sum is

$$\frac{\partial \sigma_{xx}}{\partial x}\delta x\delta y\delta z \tag{3.2.18}$$

The sum for faces with normals in the y and z directions are

$$\frac{\partial \sigma_{yx}}{\partial y}\delta x\delta y\delta z \quad \text{and} \quad \frac{\partial \sigma_{zx}}{\partial z}\delta x\delta y\delta z \tag{3.2.19}$$

Summing for all faces gives

$$\left(\frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \sigma_{yx}}{\partial y} + \frac{\partial \sigma_{zx}}{\partial z}\right)\delta x\delta y\delta z \tag{3.2.20}$$

The term σ_{xx} includes the force due to pressure, but it is directed inward and so it must be negated. So the above equation becomes

$$\left(-\frac{\partial p}{\partial t}\frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \sigma_{yx}}{\partial y} + \frac{\partial \sigma_{zx}}{\partial z}\right)\delta x\delta y\delta z \tag{3.2.21}$$

The stress tensor is defined as

$$\sigma_{xx} = 2\mu\frac{\partial u}{\partial x} \qquad \sigma_{yx} = \mu\left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}\right) \qquad \sigma_{zx} = \mu\left(\frac{\partial w}{\partial x} + \frac{\partial u}{\partial z}\right) \tag{3.2.22}$$

After combining all six faces and performing several steps of algebraic manipulation the result can be expressed as equation (3.0.1) [2].

Chapter 4

FINITE DIFFERENCE APPROACHES

One approach to solving the Navier-Stokes equations is through the use of the finite-differencing technique. This technique was utilized by [12] and [5] while a modified version was utilized by [11].

4.1 Chen and Lobo

4.1.1 Governing Equations

In [5], Chen and Lobo approach the problem of fluid simulation by solving the Navier-Stokes equations in two dimensions and then scaling the resulting pressure field to derive the height of the surface above the ground.

The equation for conservation of momentum that is used by [5] is written slightly differently than equation (3.0.1). The authors write the equation as

$$\frac{\partial \mathbf{u}}{\partial t} + u\frac{\partial \mathbf{u}}{\partial x} + v\frac{\partial \mathbf{u}}{\partial y} + \nabla p = \frac{1}{Re}\nabla^2 \mathbf{u} \tag{4.1.1}$$

where $\mathbf{u}$ and p are defined the same as equation (3.0.1) and Re is the *Reynolds number*. The Reynolds number is the ratio of inertial forces to viscous forces and is defined as $Re = \frac{\rho s L}{\mu}$ where ρ is the density, s is the characteristic velocity and L is the characteristic length of the fluid. Adjusting the Reynolds number will adjust whether or not the flow is laminar or turbulent. Flows with a small Reynolds numbers ($Re < 2100$) are considered laminar where as flows with a large Reynolds number ($Re > 4000$) are considered turbulent [30]. In true physics, the Reynolds number is a constant that is defined for different flows, however during simulation Chen and Lobo alter the Reynolds number to increase numerical stability or to achieve a desired effect.

In [5] the authors write the continuity equation as

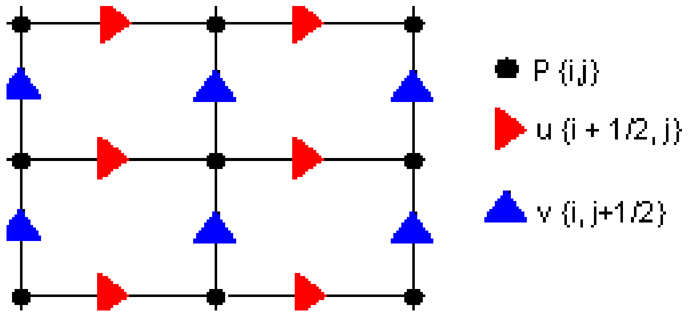
$$\epsilon p + \nabla \cdot \mathbf{u} = 0 \tag{4.1.2}$$

and note that it has been proven [27] that solving equations (4.1.1) and (4.1.2) together will tend toward the solution of equations (3.0.1) and (3.0.2) as $\epsilon \rightarrow 0$.

4.1.2 Discretization Technique

The computational domain is divided into a 2-dimensional grid where the pressure is defined at the center of each grid point and the velocities are defined at the center of the grid edges. This structure is known as the staggered marker-and-cell mesh and is illustrated in Figure 4.1.

Figure 4.1: Staggered marker-and-cell mesh



Using this grid, the velocity and the pressure are updated each frame using the following set of equations:

$$u^{n+1}_{i+1/2,j} = u^n_{i+1/2,j} + (-a^n_{i+1/2,j} - \Delta^1_x p^n_{i+1/2,j} + \frac{1}{Re}\nabla^2_h u^n_{i+1/2,j})dt \tag{4.1.3}$$

$$v^{n+1}_{i,j+1/2} = v^n_{i,j+1/2} + (-b^n_{i,j+1/2} - \Delta^1_y p^n_{i,j+1/2} + \frac{1}{Re}\nabla^2_h v^n_{i,j+1/2})dt \tag{4.1.4}$$

$$p^{n+1}_{i,j} = -\frac{\Delta^1_x u^{n+1}_{i,j} + \Delta^1_y v^{n+1}_{i,j}}{e} \tag{4.1.5}$$

where i, j indicate the coordinates of a point in the computational grid, n represents the current time step and $n+1$ represents the values after time Δt.

Δ^1_x, Δ^1_y and ∇^2_h are difference operators that are defined as

$$\Delta^1_x f_{l,m} = \frac{1}{\Delta x}(f_{l+1/2,m} - f_{l-1/2,m}) \tag{4.1.6}$$

$$\Delta^1_y f_{l,m} = \frac{1}{\Delta x}(f_{l,m+1/2} - f_{l,m-1/2}) \tag{4.1.7}$$

$$\nabla_h^2 f_{l,m} = \Delta_{xx} f_{l,m} + \Delta_{yy} f_{l,m} \tag{4.1.8}$$

$$\Delta_{xx} f_{l,m} = \frac{f_{l+1,m} - 2f_{l,m} + f_{l-1,m}}{\Delta x^2} \tag{4.1.9}$$

$$\Delta_{yy} f_{l,m} = \frac{f_{l,m+1} - 2f_{l,m} + f_{l,m-1}}{\Delta y^2} \tag{4.1.10}$$

Δ_x^1 and Δ_y^1 together compute the divergence of the pressure field (∇p from equation 4.1.1). ∇_h^2 represents the standard discretization of the Laplacian operator ($\nabla^2 \mathbf{u}$ from equation 4.1.1).

The last two symbols $a^n_{i+1/2,j}$ and $b^n_{i,j+1/2}$ are defined as

$$a^n_{i+1/2,j} = u^n_{i+1/2,j} \Delta_x^0 u^n_{i+1/2,j} + V^n_{i+1/2,j} \Delta_y^0 u^n_{i+1/2,j} \tag{4.1.11}$$

$$b^n_{i,j+1/2} = U^n_{i,j+1/2} \Delta_x^0 v^n_{i,j+1/2} + v^n_{i,j+1/2} \Delta_y^0 v^n_{i,j+1/2} \tag{4.1.12}$$

$$U_{i,j+1/2} = \frac{1}{4}(u_{i+1/2,j} + u_{i+1/2,j-1} + u_{i-1/2,j+1} + u_{i-1/2,j}) \tag{4.1.13}$$

$$V_{i+1/2,j} = \frac{1}{4}(v_{i+1,j+1/2} + v_{i,j+1/2} + v_{i,j-1/2} + v_{i+1,j-1/2}) \tag{4.1.14}$$

$$\Delta_x^0 f_{l,m} = \frac{1}{2\Delta x}(f_{l+1,m} - f_{l-1,m}) \tag{4.1.15}$$

$$\Delta_y^0 f_{l,m} = \frac{1}{2\Delta y}(f_{l,m+1} - f_{l,m-1}) \tag{4.1.16}$$

After updating the velocities and pressure using equations (4.1.3) through (4.1.5) then the velocity at each grid point can be computed as the average of its two neighbors. Namely,

$$u_{i;j} = \frac{u_{i+1/2;j} + u_{i-1/2;j}}{2} \tag{4.1.17}$$

$$v_{i;j} = \frac{v_{i;j+1/2} + v_{i;j-1/2}}{2} \tag{4.1.18}$$

4.1.3 Boundary Conditions

Boundary conditions specify the velocity of the fluid at certain "special" locations within in the grid. As the name implies, boundary conditions are used to indicate the velocity along the boundary of the computational domain. In [5] Chen and Lobo define these as external boundary conditions. Examples of external boundaries include the banks of a river or the walls of a pipe. In addition to external boundaries, Chen and Lobo define internal boundaries as boundary conditions that exist within the interior of the computational domain. These might include things such as bridges, posts or even boats.

In addition to fixed boundaries such as the banks of a river, the authors describe a method to make use of moving boundary conditions such as a boat traveling through water. Typically, during a simulation step all boundary conditions are applied and directly set the velocity of grid points lying along the boundary before the update of computational grid takes place. In the case of a moving internal boundary such as a boat, the velocity of the grid point (i, j) that lies within the boat can be set as

$$\begin{aligned} u_{i+1/2,j} &= WakeSpeed \\ u_{i-1/2,j+1} &= WakeSpeed \\ u_{i-1/2,j-1} &= WakeSpeed \end{aligned}$$

where $WakeSpeed$ is a predetermined value to represent the speed of the boat.

4.1.4 Floating Objects

In [5] the authors discuss the concept of streaklines and use them to describe the motion of objects in the flow. A streakline is defined as a collection of fluid particles that have all passed through a given point in the flow field [7]. Streaklines can be generated by releasing particles from a given point in the flow field and then tracking their position over time. Each particle is moved using the velocity of the flow field at its current location. The authors suggest treating floating objects just like streakline particles. Thus objects would be massless entities that simply assume the velocity of the flow field at any given position.

Clearly this is an extremely simple approach to objects interacting with fluid. The technique does not describe any way in which the object can affect the fluid, such as displacement or drag. Additionally, it does not account for any kind of rotational motion to be applied to the object. For these reasons it is clear that a more sophisticated approach to modeling the interaction between objects and fluid should be employed.

4.1.5 Stability

In [5] the authors do not provide a rigorous mathematical analysis of the stability of their model. They simply state that the simulation can diverge. In order to help prevent the simulation from diverging they recommend using smaller values for dt and Re and larger values for ϵ, dx, and dy. One of the advantages of simulating fluid motion in computer graphics is that one does not always need to strictly follow the laws of physics so long as the results appear correct. Therefore, in order to maintain stability a person can adjust the values of dt, dx, dy, ϵ and Re while the simulation is in progress in order to maintain stability.

Another possible method to help maintain stability would be use to Runge-Kutta (RK) integration rather than Euler integration. The obvious downside to using a more advanced integration method is the extra computation required. However, the model presented in [5] was able to achieve “interactive-rates” on hardware in 1994. Therefore, the extra computational overhead should not pose a significant problem on today’s hardware.

4.1.6 Results and Shortcomings

The authors of [5] claim to have achieved “interactive-rates” running on a Silicon Graphics Indigo. They do not provide a definition of the term “interactive-rate” but the context implies that the simulation is able to run near real-time. Theses results were achieved using the following set of values: $dt = 0.001s$, $Re = 300$, $dx = 1m$ and $dy = 1m$.

Although the model is able to run quickly and is fairly easy to implement, it does have it’s shortcomings. Given the 2D nature of the simulation it is unable to deal with objects whose cross-sectional area vary with depth. For example, the hull of a boat is curved and displaces more water as it gets deeper. The author’s fluid model simply assumes that the hull of the boat is flat along on the bottom. Additionally, since the height of the fluid is obtained by simply scaling the pressure values, it is unable to simulate overturning water. Lastly, the authors are unable to describe the exact conditions under which the simulation will remain stable.

4.2 Foster and Metaxas

4.2.1 Governing Equations of Motion

Recall that equation (3.0.1) is the general form of the Navier-Stokes equation for the conservation of momentum. The expansion of equation (3.0.1) for the u-component is

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} = -\frac{\partial p}{\partial x} + g_x + d(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{r\partial z^2}) \tag{4.2.1}$$

However, in [12] the authors have defined the u-component as

$$\frac{\partial u}{\partial t} + \frac{\partial u^2}{\partial x} + \frac{\partial uv}{\partial y} + \frac{\partial uw}{\partial z} = -\frac{\partial p}{\partial x} + g_x + d(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}) \tag{4.2.2}$$

Examining equations (4.2.1) and (4.2.2) we see that the right-hand side of each equation is identical, but the left-hand sides vary. Specifically, the question is

$$\frac{\partial u}{\partial t} + \frac{\partial u^2}{\partial x} + \frac{\partial uv}{\partial y} + \frac{\partial uw}{\partial z} \equiv \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z}$$

From a mathematical standpoint it is clear that equation (4.2.1) is not equal to equation (4.2.2). However, it will be shown later that the authors' change of notation will ultimately lead to a calculation that is similar to that presented in section 4.1.2. The equations for the v and w-components are included here for completeness.

$$\frac{\partial v}{\partial t} + \frac{\partial vu}{\partial x} + \frac{\partial v^2}{\partial y} + \frac{\partial vw}{\partial z} = -\frac{\partial p}{\partial y} + g_y + d(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial z^2})$$

$$\frac{\partial w}{\partial t} + \frac{\partial wu}{\partial x} + \frac{\partial wv}{\partial y} + \frac{\partial w^2}{\partial z} = -\frac{\partial p}{\partial y} + g_z + d(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} + \frac{\partial^2 w}{\partial z^2})$$

4.2.2 Discretization Technique

Conservation of Momentum

The discretization technique used in [12] is very similar to the technique used by [5] that was presented in section 4.1.2. The key difference is that Foster and Metaxas use a 3D grid instead of the 2D grid used in [5]. The pressure is defined at the center of the cell and the velocities are defined at the center of the cell faces (see Figure 4.2).

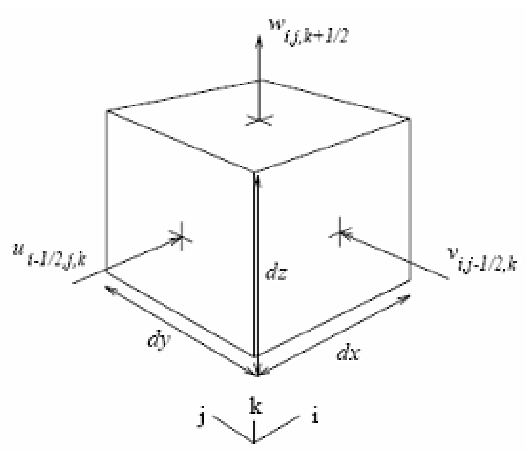


Figure 4.2: Staggered marker-and-cell grid cell. Image taken directly from [12]

The equations used to update the u-component of the velocity field are presented below. The equations for the v and w can be obtained by the obvious extension of these equations.

$$u^{n+1}_{i+1/2,j,k} = u^{n}_{i+1/2,j,k} + \frac{\partial u}{\partial t} dt \tag{4.2.3}$$

This equation takes the simulation through one full step of Euler integration.

$$\frac{\partial u}{\partial t} = -\frac{\partial u^2}{\partial x} - \frac{\partial uv}{\partial y} - \frac{\partial uw}{\partial z} - \frac{\partial p}{\partial x} + g_x + d(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}) \tag{4.2.4}$$

This equation is just a reorganization of the terms in equation (4.2.2) to explicitly show how to obtain the value of $\frac{\partial u}{\partial t}$. The equations

$$-\frac{\partial u^2}{\partial x} = \frac{1}{dx}((u_{i,j,k})^2 - (u_{i+1,j,k})^2) \ , \tag{4.2.5}$$

$$-\frac{\partial uv}{\partial y} = \frac{1}{dy}((uv)_{i+1/2,j-1/2,k} - (uv)_{i+1/2,j+1/2,k}) \tag{4.2.6}$$

and

$$-\frac{\partial uw}{\partial z} = \frac{1}{dz}((uw)_{i+1/2,j,k-1/2} - (uw)_{i+1/2,j,k+1/2}) \tag{4.2.7}$$

are simple finite-difference equations to solve for the first three terms in equation (4.2.4). The fourth term of equation (4.2.4) is expressed as:

$$-\frac{\partial p}{\partial x} = \frac{1}{dx}(p_{i,j,k} - p_{i+1,j,k}) \tag{4.2.8}$$

This too is a simple finite-difference equation to calculate the gradient of the pressure, ∇p. The last three equations,

$$d\frac{\partial^2 u}{\partial z^2} = \frac{d}{dy^2}(u_{i+1/2,j,k+1} - 2u_{i+1/2,j,k} + u_{i-1/2,j,k-1}) \tag{4.2.9}$$

$$d\frac{\partial^2 u}{\partial y^2} = \frac{d}{dy^2}(u_{i+1/2,j+1,k} - 2u_{i+1/2,j,k} + u_{i-1/2,j-1,k}) \tag{4.2.10}$$

and

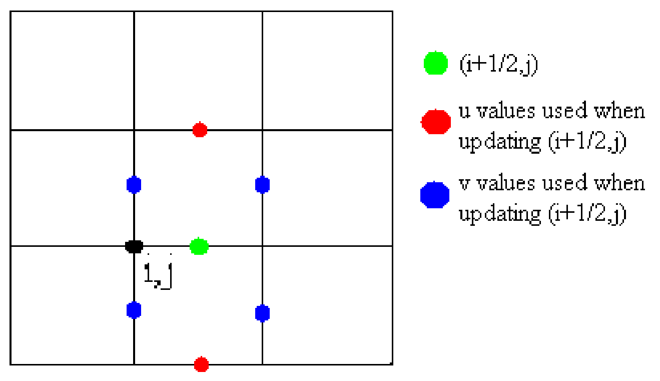
$$d\frac{\partial^2 u}{\partial x^2} = \frac{d}{dx^2}(u_{i+3/2,j,k} - 2u_{i+1/2,j,k} + u_{i-1/2,j,k}) \tag{4.2.11}$$

are a standard discretization of the Laplacian operator. For a derivation of this discretization see [**?**].

When calculating the value of $\frac{\partial u}{\partial t}$ in equation (4.2.3) it is possible that value of the velcoity field will be required at a point somewhere other than a face of grid cell. In order to remedy this situation, the average value of the nearest neighbors are used. This statment is what has allowed the authors to rewrite equation (4.2.1) as (4.2.2). As an example, assume that equation (4.2.6) is being evaluated for grid cell i, j. Figure 4.3 shows the grid points that are used to calculate the analagous term $v\partial u/\partial y$ using the method presented in section 4.1.2. Figure 4.4 shows the grid points that are used to calculate equation (4.2.6) using the method proposed by Foster and Metaxas. It is clear that many of the same grid points are utilized, however the method used by Foster and Metaxas explicity makes use of the value at $u_{i+1/2,j}$ where as the method used by Chen and Lobo does not.

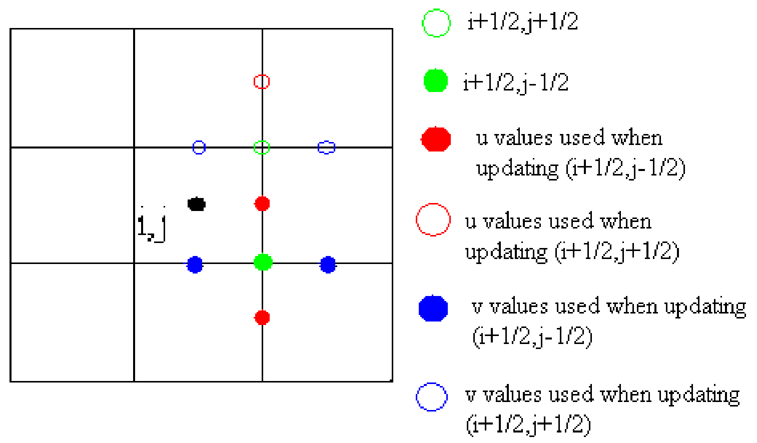
Conservation of Mass

After updating the velocity field using equation (4.2.3) it is possible that the resulting velocity field is not divergence-free and thus does not satisfy equation (3.0.2). Foster and Metaxas resolve this problem by adjusting the pressure and velocity of the cells in the computational



i,j
(i+1/2,j)
u values used when updating (i+1/2,j)
v values used when updating (i+1/2,j)

Figure 4.3: Representation of the grid points used by Chen and Lobo, [5] when calculating the term $v\partial u/\partial y$.



i,j
i+1/2,j+1/2
i+1/2,j-1/2
u values used when updating (i+1/2,j-1/2)
u values used when updating (i+1/2,j+1/2)
v values used when updating (i+1/2,j-1/2)
v values used when updating (i+1/2,j+1/2)

Figure 4.4: Representation of the grid points used by Foster and Metaxas when calculating the term $\frac{\partial uv}{\partial y}$

grid using a relaxation technique [3]. First, define the divergence, D, at cell i, j, k as

$$D_{i,j,k} = \tag{4.2.12}$$
$$-((\frac{1}{dx})(u_{i+1/2,j,k} - u_{i-1/2,j,k}) +$$
$$(\frac{1}{dy})(v_{i,j+1/2,k} - v_{i,j-1/2,k}) +$$
$$(\frac{1}{dz})(w_{i,j,k+1/2} - w_{i,j,k-1/2}))$$

Equation (4.2.12) is a central difference approximation of the divergence at cell i, j, k. Next, define the change in pressure, δp, as

$$\delta p = \frac{D_{i,j,k}\beta_0}{2dt(\frac{1}{dx^2} + \frac{1}{dy^2} + \frac{1}{dz^2})} \tag{4.2.13}$$

where β_0 is the relaxation coefficient such that $1 < \beta_0 < 2$. Next, δp is used to update the pressure of cell (i, j, k) and velocities defined on faces of cell (i, j, k).

$$u_{i+1/2,j,k} = u_{i+1/2,j,k} + (\delta t/\delta x)\delta p \qquad u_{i-1/2,j,k} = u_{i-1/2,j,k} - (\delta t/\delta x)\delta p$$
$$v_{i,j+1/2,k} = v_{i,j+1/2,k} + (\delta t/\delta y)\delta p \qquad v_{i,j-1/2,k} = v_{i,j-1/2,k} - (\delta t/\delta y)\delta p$$
$$w_{i,j,k+1/2} = w_{i,j,k+1/2} + (\delta t/\delta z)\delta p \qquad w_{i,j,k-1/2} = w_{i,j,k-1/2} - (\delta t/\delta z)\delta p$$
$$p_{i,j,k} = p_{i,j,k} + \delta p$$

After one application of the relaxation step, cell (i, j, k) will be divergence-free but changing the velocities on the cell faces may have caused neighboring cells to become divergent. Therefore, the relaxation technique must be applied to each cell many times until all cells become divergence-free. Using $\beta = 1.7$ and $\epsilon = 0.0001$ the grid was found to converge within 3 to 6 applications of the relaxation step.

4.2.3 Boundary Conditions

The model proposed by Foster and Metaxas only updates pressures and velocities to satisfy boundary conditions at the beginning of each iteration. No special calculation is used on boundary cells when updating the velocity field.

Three different types of boundary conditions are considered. The first are boundaries with stationary objects. It is assumed that faces of the objects lie on the boundary of a computational cell. Using this assumption, it becomes very easy to properly set pressure and velocity values. If an obstacle is non-permeable then the component of the velocity that is normal to the face of the obstacle is set to zero ($u_0 = 0$). This will prevent water from penetrating the face. If the no-slip boundary condition, where the obstacle creates a drag on the fluid, is desired then the component of the velocity field that is tangential to the face of the object is set to zero as well. However, since the face of an obstacle can not lie on a cell boundary (see Figure 4.5) where both u and v are defined, then the tangential velocity is indirectly set to zero by setting the velocity on the edge inside of the cell equal to the opposite of the velocity outside of the obstacle ($v_0 = -v_1$). Lastly, the pressure inside of the cell containing the obstacle is set to the pressure of the neighboring cell outside of the obstacle ($p_0 = p_1$). If a free-slip boundary is desired then the velocities and pressures are set the same way as for a no-slip boundary except that the tangential velocity inside the obstacle is set equal to the tangential velocity outside of the obstacle ($v_0 = v_1$). This will prevent any acceleration across the boundary.

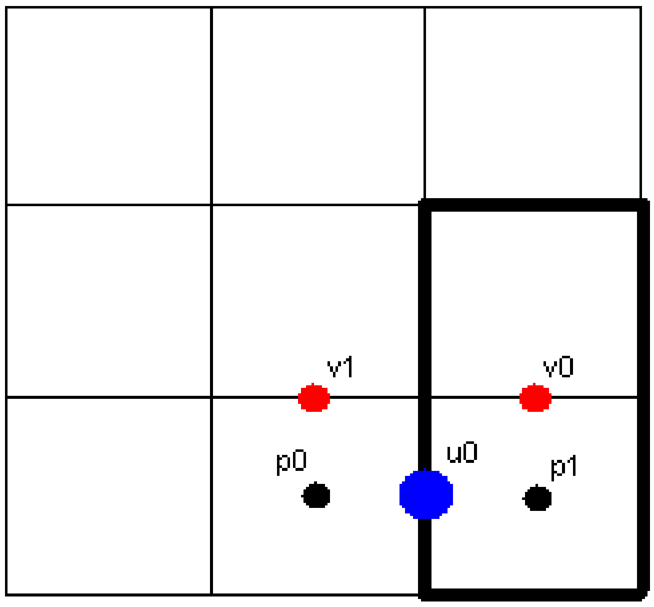


Figure 4.5: An obstacle in the computational grid. The obstacle is denoted by the thick black lines.

Inflow and outflow boundaries are the second type of boundary condition to consider. Inflow boundaries allow water to enter into the system and are implemented by setting the face velocities of an inflow cell equal to a predetermined value and then holding it fixed for the entire duration of the simulation. Outflow cells have their inital velocity set to that of their neighbors. Then during the iterative relaxation technique used to update pressure values, the velocities are allowed to change without constraint.

The third and final boundary condition to consider is the boundary at the surface between the water and the air. Each cell in the mesh is marked as either *Full*, *Surface* or *Empty* using one of the surface tracking techniques described in section 4.2.4. For any cell that has been marked as *Surface* the velocity at the faces opposite of a *Full* cell are set so that the divergence of the *Surface* is zero. The pressure in the *Surface* cell is set to the atmospheric pressure.

4.2.4 Surface Tracking

Three different surface tracking techniques are introduced and each is applicable to different requirements. The first technique utilizes marker particles. Marker particles are massless and are introduced at inflow boundaries. The particles are updated by sampling the value of the fluid velocity field at the particles position, $\mathbf{x}_p$ and then multiplying by the current timestep. Cells in the computational mesh are then labeled in the following fashion:

- A cell devoid of particles is marked as *Empty*
- A cell that contains at least one particle and is next to a cell that has been labled as *Empty* is labeled as *Surface*.
- A cell that has at least one particle and is not labeled as a *Surface* cell is then labeled as *Full*.

Marker partilces are useful in situations where the water is likely to exhibit violent phenomena. For example, Figure 4.6 shows water that has been poured into a tank and turned over on itself.

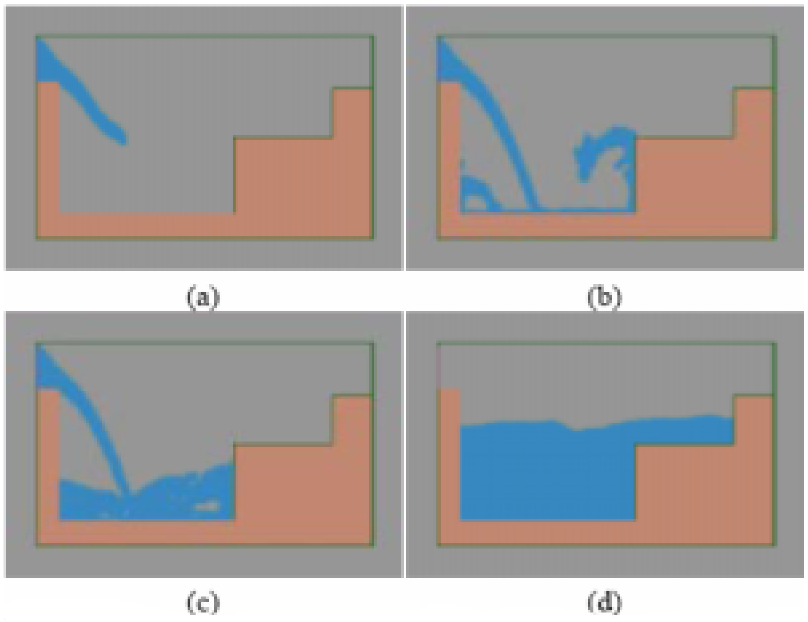


Figure 4.6: Frames from a 2D animation that utilizes marker particles. Image is reproduced from [12].

The second method used to track the surface makes use of Free Surface Particles. These too are massless particles that are advected with the fluid. The difference is that instead of moving throughout the entire fluid volume, these particles are located only at the boundaries between fluid and obstacles or air. If two particles are too close together, then they are deleted and their neighbors connected together. If two particles become too far apart, a new particle is introduced on the link between the two. For each iteration of the simulation the cells are labeled as *Full*, *Surface* and *Empty* using a region-growing algorithm as follows:

- Change current *Surface* cells into *Full* cells.
- Start from a cell that is known to be *Empty* for the duration of the simulation and grow a region of *Empty* cells until a *Full* cell is encountered.
- Grow a region of *Full* cells until either a boundary or *Empty* cell is encountered.
- Grow *Empty* cells again and repeat alternating *Empty* and *Full* whenever a boundary is encountered.
- Set all original *Surface* cells back to *Surface*

The third and final surface tracking method uses a heightfield approach. Instead of just scaling the pressure at each grid cell as done by Chen and Lobo in [5] the underlying fluid velocities are considered. The height of the surface above the terrain is defined along the y axis passing through each vertical column of cells. The general formula used to update the surface height, h, is:

$$\frac{\partial h}{\partial t} = w - u(\frac{\partial h}{\partial t}) - v(\frac{\partial h}{\partial y}) \tag{4.2.14}$$

Equation (4.2.14) is discretized as:

$$\begin{aligned} h_{i,j}^{t+\delta t} &= h_{i,j}^{t} + \delta t(w_{i,j,k}^{t+\delta t} \\ &+ \frac{(h_{i-1,j}^{t} - h_{i+1,j}^{t})}{4\delta x}(u_{i+1/2,j,k}^{t+\delta t} + u_{i-1/2,j,k}^{t+\delta t}) \\ &+ \frac{(h_{i-1,j}^{t} - h_{i+1,j}^{t})}{4\delta y}(v_{i,j+1/2,k}^{t+\delta t} + u_{i,j-1/2,k}^{t+\delta t})) \end{aligned}$$

Once the height of each column has been determined, then the process of labeling cells is trivial. Any cell above the heightfield is labeled as *Empty*, cells below the heightfield are marked as *Full* and cells that intersect the heightfield are labeled as *Surface*.

4.2.5 Rigid Bodies

The authors describe a model that can be used to simulate the forces exerted on a rigid body by the fluid. They do not however, describe a method to simulate the forces that a rigid body exerts on the fluid. A rigid body, B is divided into a set of n nodes. For each node within the fluid the force acting upon it can be calculated as:

$$\mathbf{f}_{n_i} = -\nabla p_i dV_i + m_i \mathbf{g} \tag{4.2.15}$$

where dV_i is the volume of the submerged node. The term, ∇p_i, is the gradient of the pressure and is defined as:

$$(\nabla p_i)_{x_j} = \frac{p_{n_i} - p_{n_i,x_j}}{\delta x} \tag{4.2.16}$$

where p_{n_i} is the pressure in the cell containing n_i and p_{n_i,x_j} is the pressure in the previous cell in the x direction [12]. $\mathbf{g}$ is acceleration due to gravity and m_i is the mass of node i. The total force acting on the object is the sum of the force acting on each node.

4.2.6 Results

All results were obtained on a Silicon Graphics Crimson R4000 running at either 100MHz or 150MHz [15]. An animation entitled *Moonlight Cove* was generated on a 50x15x40 mesh. The simulation completed 20,000 iterations in two and half hours. Total time includes rendering that was done using RenderMan. This equates to roughly two iterations per second. Another example was computed on a 40x12x40 grid. This was able to complete 8,000 iterations in one hour. In this example rendering was done using “standard Silicon Graphics hardware routines” [12]. This also equates to roughly 2 iterations per second.

Chapter 5

STABLE METHODS

Stable methods consist of solving the Navier-Stokes equations in a form that permits an arbitrarily large time step to be taken without the risk of the simulation becoming unstable. Jos Stam's seminal paper entitled simply "Stable Fluids" will act as the foundation of the two implicit methods described herein. The other method by Song et al is a direct extension to Stam's method which is intended to overcome some of the known limitations of Stam's model.

5.1 *Stable Fluids by Jos Stam*

5.1.1 Governing Equations of Motion

The equations of motion used by this model are the same as equations (3.0.1) and (3.0.2). They are reproduced here for convenience:

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla)\mathbf{u} - \frac{1}{\rho}(\nabla p) + \nu \nabla^2 \mathbf{u} + \mathbf{f} \tag{5.1.1}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{5.1.2}$$

Stam makes use of the *Helmholtz-Hodge Decomposition* from linear algebra to connect equations (5.1.1) and (5.1.2) into one equation.

5.1.2 Helmholtz-Hodge Decomposition

The Helmholtz-Hodge decomposition theorem states that any vector field, $\mathbf{w}$, can be decomposed into two parts, such that:

$$\mathbf{w} = \mathbf{u} + \nabla p \tag{5.1.3}$$

where $\nabla \cdot \mathbf{u} = 0$ and p is a scalar field. In English, this means that any vector field $\mathbf{w}$ can be decomposed into the sum of a divergence-free vector field, $\mathbf{u}$, and the gradient of a scalar field,

∇p. It can also be shown that $\mathbf{u} = \mathbf{w} - \nabla p$. From this definition we define an operator, $\mathbf{P}$, that will project any vector field onto its divergence free component, $\mathbf{u} = \mathbf{P}\mathbf{w}$. Applying the projection operator, $\mathbf{P}$ to both sides of equation (5.1.3) yields:

$$\begin{aligned} \mathbf{P}\mathbf{w} &= \mathbf{P}(\mathbf{u}) + \mathbf{P}(\nabla p) \\ & \quad \mathbf{u} + \mathbf{P}(\nabla p) \text{ Divergence-free part of } \mathbf{u} \text{ is } \mathbf{u} \end{aligned}$$

Since we know that the divergence-free part of $\mathbf{w}$ is $\mathbf{u}$ then $\mathbf{P}(\mathbf{w}) = \mathbf{u}$ and therefore $\mathbf{P}(\nabla p) = 0$ must be true.

Applying the projection operator $\mathbf{P}$ to both sides of equation (5.1.1) we get a single equation that be can be used to update the velocity field and then project it onto its divergence-free part.

$$\mathbf{P}(\frac{\partial \mathbf{u}}{\partial t}) = \mathbf{P}(-(\mathbf{u} \cdot \nabla)\mathbf{u} - \frac{1}{\rho}(\nabla p) + \nu \nabla^2 \mathbf{u} + \mathbf{f}) \tag{5.1.4}$$

Since $\mathbf{u}$ is already divergence-free then:

$$\frac{\partial \mathbf{u}}{\partial t} = \mathbf{P}(-(\mathbf{u} \cdot \nabla)\mathbf{u} - \frac{1}{\rho}(\nabla p) + \nu \nabla^2 \mathbf{u} + \mathbf{f}) \tag{5.1.5}$$

Lastly, since $\mathbf{P}(\nabla p) = 0$ then the final equation is:

$$\frac{\partial \mathbf{u}}{\partial t} = \mathbf{P}(-(\mathbf{u} \cdot \nabla)\mathbf{u} - \nu \nabla^2 \mathbf{u} + \mathbf{f}) \tag{5.1.6}$$

The following sections will discuss the step-by-step process used to solve each term on the right-hand side of equation (5.1.6) and then how to project the resultant velocity field onto its divergence-free component in order to satisfy equation (5.1.2).

5.1.3 Four-Step Fluid Solver

The solver proceeds in four-steps. The resultant vector field after the application of the current step will become the input to the next step. It is important to note that the grid used by Stam defines the both the pressure and the velocity at the *center* of each computational cell. The finite-difference approaches discussed above only defined the pressure at the center and the velocities were defined on the cell faces.

$$\mathbf{w_0}(\mathbf{x}) \overset{\text{add force}}{\rightarrow} \mathbf{w_1}(\mathbf{x}) \overset{\text{advect}}{\rightarrow} \mathbf{w_2}(\mathbf{x}) \overset{\text{diffuse}}{\rightarrow} \mathbf{w_3}(\mathbf{x}) \overset{\text{project}}{\rightarrow} \mathbf{w_4}(\mathbf{x})$$

External Forces

$$\mathbf{w_0}(\mathbf{x}) \overset{\text{add force}}{\rightarrow} \mathbf{w_1}(\mathbf{x})$$

The easiest step to update is the addition of external forces. In this case:

$$\mathbf{w}_1(\mathbf{x}) = \mathbf{w_0}(\mathbf{x}) + \Delta t \mathbf{f}(\mathbf{x}, t) \tag{5.1.7}$$

where Δt is the time step and $\mathbf{f}(\mathbf{x}, t)$ is the vector field representing the external forces.

Advection

$$\mathbf{w_0}(\mathbf{x}) \overset{\text{add force}}{\rightarrow} \mathbf{w_1}(\mathbf{x}) \overset{\text{advect}}{\rightarrow} \mathbf{w_2}(\mathbf{x})$$

The advection term of the Navier-Stokes equation, $((\mathbf{u} \cdot \nabla)\mathbf{u})$, is solved using a technique known as the *Method of Characteristics*. The method of characteristics can be used to solve advection equations of the type

$$a\frac{\partial u}{\partial x} + \frac{\partial u}{\partial t} = 0 \ . \tag{5.1.8}$$

Using the method of characteristics our goal is to rewrite the partial differential equation in the form of an ordinary differential equation over some characteristic curve parameterized by s.

$$x(s), t(s) \tag{5.1.9}$$

is the characteristic curve. Using the chain rule of differentiation shows that:

$$\frac{d}{ds}u(x(s), t(s)) = \frac{dx}{ds}\frac{\partial u}{\partial x} + \frac{dt}{ds}\frac{\partial u}{\partial t}. \tag{5.1.10}$$

Notice that if we set $\frac{dx}{ds} = u$ and $\frac{dt}{ds} = 1$ then we get:

$$\frac{d}{ds}u(x(s), t(s)) = u\frac{\partial u}{\partial x} + \frac{\partial u}{\partial t} \tag{5.1.11}$$

which are the first two terms of equation (4.2.1) and are in the same form as equation (5.1.8) with $a = u$. From this we can write:

$$\frac{d}{ds}u(x(s), t(s)) = u\frac{\partial u}{\partial x} + \frac{\partial u}{\partial t} = 0 \tag{5.1.12}$$

Now we have three seperate ODEs to solve.

$$\frac{dt}{ds} = 1$$

With the initial condition that $t(0) = 0$ then we know that $t = s$.

$$\frac{dx}{ds} = u$$

with the initial condition that $x(0) = x_0$ we know that $x = us + x_0 = ut + x_0$.

$$\frac{du}{ds} = 0 \tag{5.1.13}$$

with the initial condition $u(0) = f(x_0)$ then $u(t) = u(x_0)$. Therefore

$$u(x, t) = u(x - ut) \tag{5.1.14}$$

In English, equation (5.1.14) states that for our situation, the velocity at point x is equal to the velocity at position $x - u\delta t$.

When solving the advection portion of the Navier-Stokes equation we will examine each point in the computational grid. Using the velocity at that point we will trace the point backwards through the flow field to the point that it was at during the last iteration. Since the backtraced point will rarely land directly on a computational point, then the velocity values are linearly interpolated from the nearest neighbors. The velocity at the current update point is then set to the interpolated value at the backtraced point. See figure 5.1 for an overview of the solver up to this point.

This technique is sometimes also referred to as a *semi-Lagrangian* technique. It is this step of the solver that allows it to become unconditionally stable. No matter how large the time step

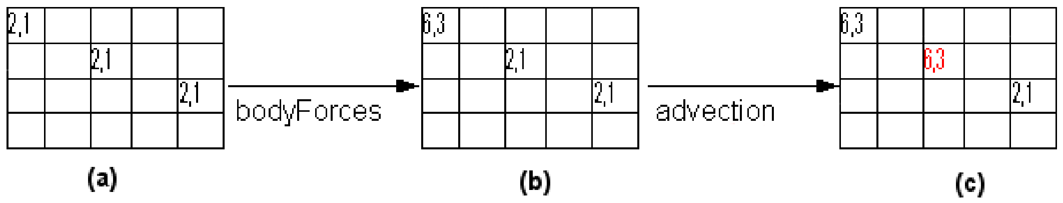


Figure 5.1: (a) The original velocity field. The integer pair represents the velocity in the x and y directions respectively. The field is shown in 2D and only specific cells are explicitly defined simply for ease of representation. (b) The velocity field after the application of external forces. (c) The resultant velocity field after the update due to advection. It's important to note that the value in red is the value of w2, not w1, thus not affecting the last explicitly cell until the next iteration of the solver.

the values of the velcoity field will never be smaller or larger than they were in the previous time step unless external forces have acted upon them.

Diffusion

$$\mathbf{w_0}(\mathbf{x}) \overset{\text{add force}}{\longrightarrow} \mathbf{w_1}(\mathbf{x}) \overset{\text{advect}}{\longrightarrow} \mathbf{w_2}(\mathbf{x}) \overset{\text{diffuse}}{\longrightarrow} \mathbf{w_3}(\mathbf{x}) \tag{5.1.15}$$

Fick's second law of diffusion states [28]

$$\frac{\partial \mathbf{w_2}}{\partial t} = \nu \nabla^2 \mathbf{w_2} \tag{5.1.16}$$

where ν is the kinematic viscosity. We could use an explicit formulation to derive the velocity field w_3 from w_2 by writing

$$\mathbf{w_3} = \mathbf{w_2} + (\nu \nabla^2 \mathbf{w_2})\delta t \tag{5.1.17}$$

however, this could cause the simulation to become unstable if the viscosity is large ([26]). Instead, an implicit formulation is used

$$(\mathbf{I} - \nu \delta t \nabla^2)\mathbf{w_3} = \mathbf{w_2} \tag{5.1.18}$$

where $\mathbf{I}$ is the identity. This is a Poisson equation that can be solved using a number of different techniques. We are going to use the Gauss-Seidel Relaxation technique. The same relaxation technique is used by the projection step. See section 5.1.3 for a description.

Projection

$$\mathbf{w_0}(\mathbf{x}) \overset{\text{add force}}{\longrightarrow} \mathbf{w_1}(\mathbf{x}) \overset{\text{advect}}{\longrightarrow} \mathbf{w_2}(\mathbf{x}) \overset{\text{diffuse}}{\longrightarrow} \mathbf{w_3}(\mathbf{x}) \overset{\text{project}}{\longrightarrow} \mathbf{w_4}(\mathbf{x})$$

We know that the vector field w_3 can be decomposed into the sum of a divergence-free vector field and the gradient of a scalar field

$$\mathbf{w_3} = \mathbf{u} + \nabla p \ . \tag{5.1.19}$$

Applying the divergence operator to both sides of equation (5.1.19) yields

$$\nabla \cdot \mathbf{w_3} \quad = \quad \nabla \cdot \mathbf{u} + \nabla \cdot \nabla p \tag{5.1.20}$$

$$\nabla^2 p \quad \text{(since } \mathbf{u} \text{ is divergence-free then } \nabla \cdot \mathbf{u} = 0). \tag{5.1.21}$$

This is another Poisson equation that is similar in form to equation (5.1.18). When implementing the projection step we will solve equation (5.1.20) for p and then subtract its gradient to obtain the divergence-free field.

$$\mathbf{w_4} = \mathbf{w_3} - \nabla p \tag{5.1.22}$$

Solving Poisson equations

The Laplacian operator is defined in two dimensions as

$$\nabla^2 p = \frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} \ . \tag{5.1.23}$$

When discretized, each term above becomes

$$\frac{\partial^2 p}{\partial x^2} = \frac{1}{\partial x^2}(p_{i-1,j} + p_{i+1,j} - 2p_{i,j}) \tag{5.1.24}$$

$$\frac{\partial^2 p}{\partial y^2} = \frac{1}{\partial y^2}(p_{i,j-1} + p_{i,j+1} - 2p_{i,j}) \tag{5.1.25}$$

where δx and δy represent the grid spacing in the x and y directions respectively. If we assume that $\delta x = \delta y$ then the equations simplify to

$$\nabla^2 p = \frac{1}{\delta x^2}(p_{i-1,j} + p_{i+1,j} + p_{i,j-1} + p_{i,j+1} - 4p_{i,j}) \ . \tag{5.1.26}$$

As an example, let's solve the Poisson equaton, (5.1.18), from section 5.1.3.

$$\begin{aligned}
\mathbf{w_2} &= (\mathbf{I} - \nu\delta t\nabla^2)\mathbf{w_3} \\
&= -\nu\delta t\nabla^2\mathbf{w_3} + \mathbf{I}\mathbf{w_3} \quad \text{Distribute } w_3 \\
&= -\nu\delta t(\frac{\mathbf{w_3}_{i-1,j} + \mathbf{w_3}_{i+1,j} + \mathbf{w_3}_{i,j-1} + \mathbf{w_3}_{i,j+1} - 4\mathbf{w_3}_{i,j}}{\delta x^2}) + \mathbf{w_3}_{i,j} \quad \text{Discretize } \nabla^2 \\
&= (\frac{\mathbf{w_3}_{i-1,j} + \mathbf{w_3}_{i+1,j} + \mathbf{w_3}_{i,j-1} + \mathbf{w_3}_{i,j+1} - 4\mathbf{w_3}_{i,j}}{\delta x^2/-\nu\delta t}) + \mathbf{w_3}_{i,j}\frac{\delta x^2/-\nu\delta t}{\delta x^2/-\nu\delta t} \\
&= (\frac{\mathbf{w_3}_{i-1,j} + \mathbf{w_3}_{i+1,j} + \mathbf{w_3}_{i,j-1} + \mathbf{w_3}_{i,j+1} - 4\mathbf{w_3}_{i,j} + \mathbf{w_3}_{i,j}(\delta x^2/-\nu\delta t)}{\delta x^2/-\nu\delta t})
\end{aligned}$$

$$\mathbf{w_2}(\delta x^2/-\nu\delta t) = \mathbf{w_3}_{i-1,j} + \mathbf{w_3}_{i+1,j} + \mathbf{w_3}_{i,j-1} + \mathbf{w_3}_{i,j+1} - \mathbf{w_3}_{i,j}(4 - (\delta x^2/-\nu\delta t)) \tag{5.1.27}$$

Here we have a linear system of equations that can be represented by the equation $\mathbf{Ax} = \mathbf{b}$ where $\mathbf{b} = \mathbf{w_2}(\delta x^2/-\nu\delta t)$ is the known vector field, $\mathbf{x} = \mathbf{w_3}$ is the unknown vector field and $\mathbf{A}$ is a sparse matrix whose diagonal elements are $(4 - (\delta x^2/-\nu\delta t))$ and whose off-diagonal elements represent the neighbors of grid cell (i, j) and are set to one. This sparse linear system can be solved many different ways. We will choose to solve it using Gauss-Seidel Relaxation. The derivation for the Poisson equation in the pressure projection step follows a very similar and simpler sequence as the one presented here.

Gauss-Seidel Relaxation

Gauss-Seidel Relaxation (GSR) is an iterative technique intended to solve systems of linear equations of the form $\mathbf{Ax} = \mathbf{b}$ where $\mathbf{A}$ is a matrix, $\mathbf{x}$ is a vector of unknown values and $\mathbf{b}$ is a vector of known values. The formula for GSR is [1]

$$\mathbf{x}_i^k = \frac{1}{a_{i,i}}[b_i - \sum_{j=1}^{i-1} a_{i,j}x_j^k - \sum_{j=i+1}^{N} a_{i,j}x_j^{k-1}] \tag{5.1.28}$$

where $\mathbf{x}^k$ is the value of $\mathbf{x}$ at the k^{th} iteration. Using the values of $\mathbf{A}$, $\mathbf{x}$, and $\mathbf{b}$ described at the end of section 5.1.3 we obtain the final equation for the velocity field $\mathbf{w_3}$ as

$$\mathbf{w_3}^k_{i,j} = \frac{1}{4-(\delta x^2/-\nu\delta t)}[\mathbf{w_2}_{i,j}(\delta x^2/-\nu\delta t) - \mathbf{w_3}_{i-1,j} - \mathbf{w_3}_{i,j-1} - \mathbf{w_3}_{i+1,j} - \mathbf{w_3}_{i,j+1}] \quad (5.1.29)$$

The final equation for the k^{th} iteration of the pressure field is included here for completeness

$$p^k_i = \frac{1}{4}[\nabla \cdot \mathbf{w}_{3_{i,j}}(\delta x^2) - p^k_{i-1,j} - p^k_{i,j-1} - p^{k-1}_{i+1,j} - p^{k-1}_{i,j+1}]\ . \quad (5.1.30)$$

5.2 Song, Shin and Ko

The model developed by Song et al in [25] is directly derived from the model developed by Stam in [26] and discussed in section 5.1. The primary contributions made by Song et. al are the introduction of the level set method to track the surface of the water, the use of the Constrained Interpolation Profile method during the semi-Lagrangian advection, description of modeling air bubbles and water droplets and a method to simulate the interaction between the water and rigid bodies.

5.2.1 Level Set Methods

Level set methods were introduced by Sethian and are applicable to a wide variety of problems. In fluid simulation they are frequently used to track the free-surface between two fluids. In this case, it is the surface between water and air. For a good introduction to level sets see [24].

To track the free surface between the water and the air we use a signed distance function, ϕ. The value of the function is defined to be positive at all points inside of the water and negative outside of the water and is tracked at the center of each computational grid cell. The position of the free-surface is defined at all points where $\phi = 0$. When the simulation begins each grid cell is initialized with its distance from the surface. As the simulation progresses the distance is updated using the level set equation [16]

$$\frac{\partial \phi}{\partial t} + \mathbf{u} \cdot \nabla \phi = 0 \quad (5.2.1)$$

The level set equation looks very similar to the advection portion of the Navier-Stokes equation

(3.0.1) and the same technique can be used to compute both values.

5.2.2 The Constrained Interpolation Profile Method

One of the problems with the Stable Fluids model developed by Stam [26] is that the model can lead to a dissapation of mass. Song et al propose a solution based on the use of the Constrained Interpolation Profile (CIP) method.

In Stam's model, when solving for the advection portion of the Navier-Stokes equation a semi-Lagrangian technique is used. See section 5.1.3 for a description. In Stam's method after the particle has been backtraced then linear interpolation is used to compute the velocity based on nearest neighbors. Song et. al instead use a cubic interpolation method that can help recover sub-cell features that might be lost by using simple linear interpolation.

The CIP method makes use of the value at neighboring grid cells as well as the spatial derivatives of those values. The derivative of the values can be obtained by directly differentiating the original advection equation. In the case of the level set equation

$$\frac{\partial \phi_\xi}{\partial t} + \mathbf{u} \cdot \nabla \phi_\xi = -\mathbf{u}_\xi \cdot \nabla \phi \tag{5.2.2}$$

where $\phi_\xi = \frac{\partial \phi}{\partial \xi}$, $\mathbf{u}_\xi = \frac{\partial \mathbf{u}}{\partial \xi}$ and ξ is one of the spatial variables.

In one dimension the CIP equation is written as

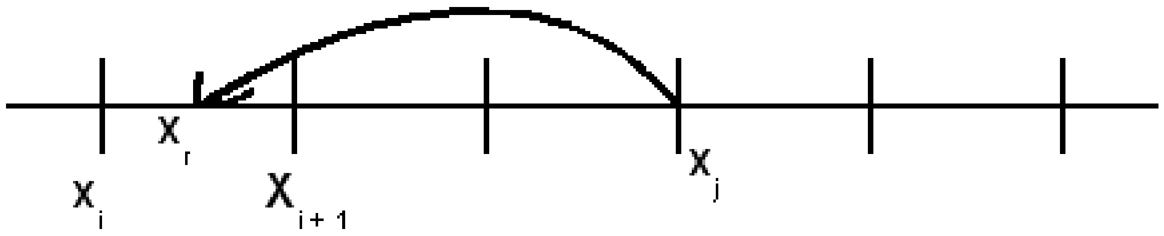
$$\Phi(\mathbf{X}) = a\mathbf{X}^3 + b\mathbf{X}^2 + \acute{\phi}_i \mathbf{X} + \phi_i \tag{5.2.3}$$

where $\mathbf{X} = \mathbf{x} - \mathbf{x}_i$ for $\mathbf{x} \in [\mathbf{x}_i, \mathbf{x}_{i+1}]$ and

$$\begin{aligned} a &= \frac{\acute{\phi}_i + \acute{\phi}_{i+1}}{\Delta x^2} - \frac{2\Delta\phi}{\Delta x^3} && (5.2.4)\\ b &= \frac{3\Delta\phi}{\Delta x^2} - \frac{2\acute{\phi}_i + \acute{\phi}_{i+1}}{\Delta x} && (5.2.5)\\ \Delta\phi &= \phi_{i+1} - \phi_i && (5.2.6)\\ \Delta x &= x_{i+1} - x_i && (5.2.7) \end{aligned}$$

As an example, suppose that x_j is a one dimensional point and it has been backtraced to $x_r \in [x_i, x_{i+1}]$. See figure 5.2. Then $X_r = x_r - x_{i+1}$, $\phi_j = \Phi(X_r)$ and $\acute{\phi} = \acute{\Phi}(X_r)$.

Figure 5.2: A one-dimensional example of the CIP method

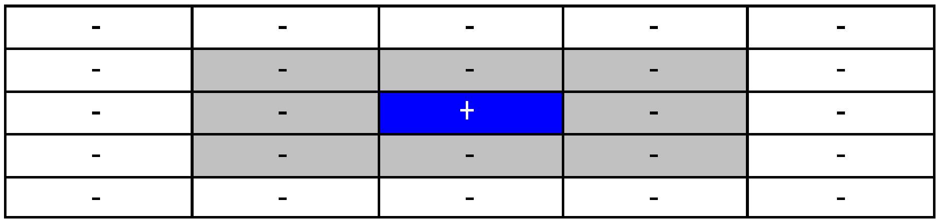


5.2.3 *Air Bubbles and Water Droplets*

The process of simulating bubbles and droplets can be divided into four parts. The first is identifying when and where bubbles and droplets should be created. Once the location has been identified then the volume represented by the droplet/bubble needs to be determined. The third is moving the bubbles or droplets. The last is merging bubbles or droplets back into the main fluid simulation.

Identifying Bubbles and Droplets

Song et. al describe three methods that can be used to identify air bubbles and water droplets. The first method involves examining the sign of the level set function. If cell with a positive level set value is surrounded by cells with a negative level set value then a droplet is introduced. If the signs were reversed then a bubble is created. See Figure 5.3.



-	-	-	-	-
-	-	-	-	-
-	-	+	-	-
-	-	-	-	-
-	-	-	-	-

Figure 5.3: The first method to identify water droplets. In this scenario a water droplet would be created at the position of the blue cell.

Sometimes, simply examining the signs of the level set function is not enough. Consider the case where the values inside a of small grid have a small negative value and are surrounded by cells with a large negative value. The small negative values indicate that water is nearby so a water droplet should be introduced into the blue region in figure 5.4.

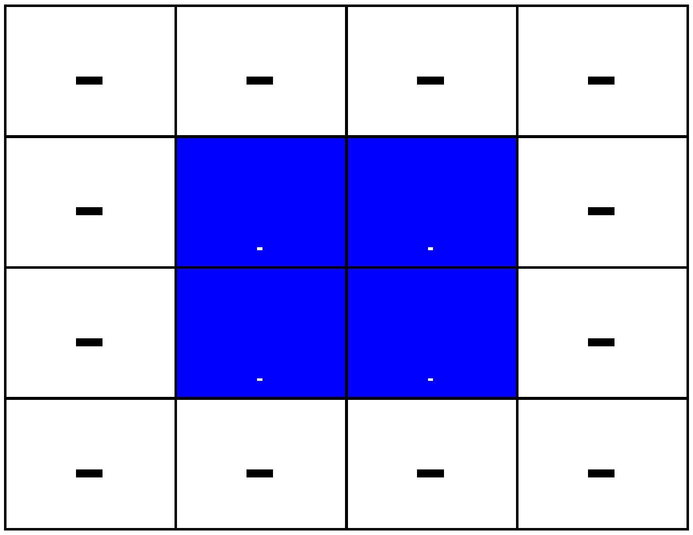


Figure 5.4: The second method to identify water droplets. The blue cells contain a small negative level set value while the surrounding cells contain a large negative level set value.

Lastly, the semi-Lagrangian technique used in the advection step of the solver may not always refer to every cell in the grid. The mass in the unreferred cells may be lost. To counteract this problem those cells are converted into droplets/bubbles.

Calculating the Volume of Droplets and Bubbles

Once the position of droplets and bubbles has been identified then the volume of fluid to be represented by the droplet/bubble needs to be determined. The authors use the following equation

$$V_f = \int_{c} H(\phi(\mathbf{x}))d\mathbf{x} \approx \sum_{c} H_\epsilon(\phi(\mathbf{x}_c))\Delta x \Delta y \Delta z \tag{5.2.8}$$

where $_c$ represents the droplet (circular region), c is the index over the cells in the shaded region of figure 5.5 and H is the smeared Heaviside function given by [25]

$$H_\epsilon(\phi) = \begin{cases} 0 & \text{if } \phi < -\epsilon \\ \frac{1}{2} + \frac{\phi}{2\epsilon} + \frac{1}{2\pi}\sin(\frac{\phi\pi}{\epsilon}) & \text{if } |\phi| \leq \epsilon \\ 1 & \text{if } \phi > \epsilon \end{cases} \tag{5.2.9}$$

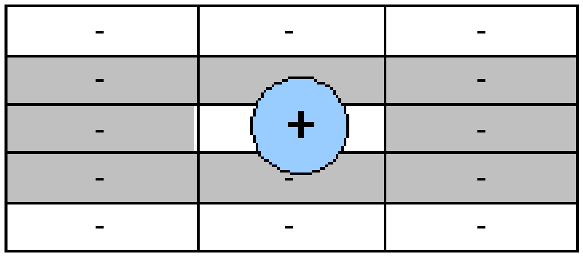


Figure 5.5: The shaded cells are used to determine the volume of the droplet.

Motion of Droplets and Bubbles

Bubbles and droplets can be thought of as simply particles. As the particle moves it experiences forces due to gravity, drag and pressure from the surrounding fluid. The authors summarize the force as

$$\mathbf{f} = m_f \mathbf{g} + \alpha_d r^2 (\mathbf{b} - \mathbf{v}_p)|\mathbf{b} - \mathbf{v}_p| - V_f \nabla p \tag{5.2.10}$$

where m_f is the mass, V_f is the volume, α_d is the drag coefficient, r is the radius of the particle, $\mathbf{v}_p$ is the current velocity of the particle and $\mathbf{b}$ is the interpolated velocity of the grid-based fluid measured at the center of the particle. The last two terms of the equation represent the force of the fluid acting on the particle and must be accompanied by an opposite force acting on the fluid.

Merging Droplets/Bubbles Back Into the Fluid

Particles are merged back into the fluid when one of the two following conditions are met:

1. The volume of the particle becomes more than twice as large as a single cell.

2. The particle hits the surface or moves into portion of the fluid of the same phase (droplet moves into water or bubble moves into air).

In the first case, the velocity at the grid points is set to the velocity of the particle and the level set value is updated according to

$$\phi(\mathbf{x}_i) = s_p(\mathbf{r}_p - |\mathbf{x}_i - \mathbf{x}_p|) \tag{5.2.11}$$

where $s_p = 1$ for a droplet and $s_p = -1$ for a bubble, r_p is the radius of the particle, $\mathbf{x}_p$ is the center of the particle and $\mathbf{x}_i$ is the grid point being updated.

In the second case where the particle moves back into the fluid of the same phase, then the velocity at the grid point is set to the average of the current velocity and the particle's velocity. The level set values are updated using the inverse of the procedure that was used to determine the volume of the particle. This procedure has the effect of forming small ripples. When the droplet hits the surface it creates a small downward force creating a hollow. At later time steps the region is pushed back up and creates a small bump.

5.2.4 Interacting with Rigid Bodies

When a rigid object enters the water, the cells whose centers are contained within the object are marked. For each marked cell, s, the component of the velocity field that is normal to the object must be constrained to be positive or else water will be flowing into the object. Therefore, the velocity of the cell is modified so that

$$\mathbf{u}_s \cdot \mathbf{n}_s \geq 0 \; . \tag{5.2.12}$$

To ensure this equality holds, the normal component of $\mathbf{u}$ is removed and the tangential component stays the same. The linear and angular forces on the object can be described as:

$$\mathbf{F} = M\mathbf{g} + \sum_s (-\nabla p_s \cdot \mathbf{n}_s)\mathbf{n}_s \Delta S \tag{5.2.13}$$

$$\mathbf{T} = \sum_s (\mathbf{r}_s - \mathbf{r}_c) \times (-\nabla p_s \cdot \mathbf{n}_s)\mathbf{n}_s \Delta S \tag{5.2.14}$$

Where $\mathbf{F}$ is the total linear force, $\mathbf{T}$ is total torque, M is the mass of the object, s is the index ranging over the marked cells, p_s is the fluid pressure of the cell, $\mathbf{r}_s$ is the position of the cell, $\mathbf{r}_c$ is the position of the object's center of mass and ΔS is the area of the object surface subsumed in the cell.

5.2.5 *Results*

Using a 3.2 GHz Intel Pentium 4 with 1 GB of memory the authors were able to achieve real-time results (30-40 fps) in 2D. Simulation times in 3D ranged from 34.0 to 51.7 seconds per frame on an 80x80x80 grid.

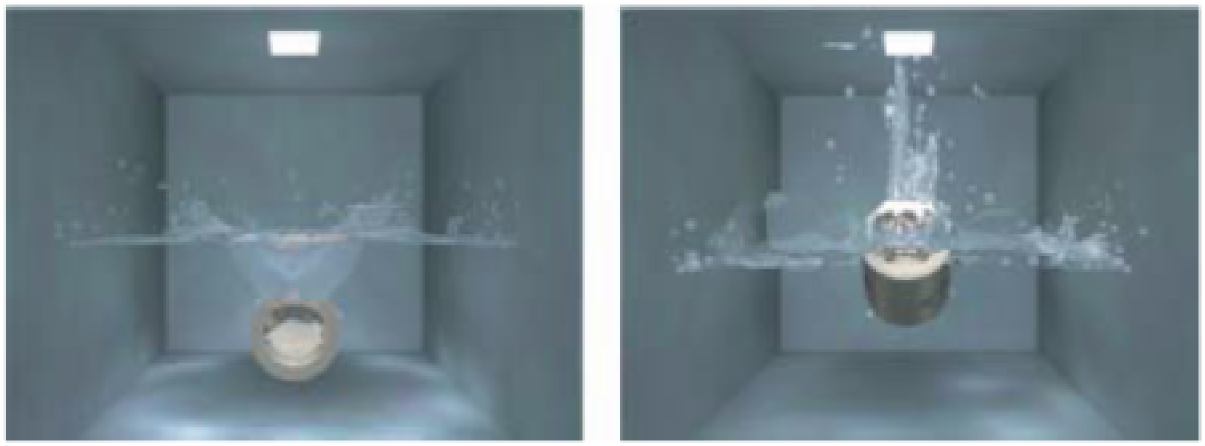


Figure 5.6: A 3D simulation of a cup being drowned in water using the fluid simulation model developed by Song et. al. Image taken directly from [25]

Chapter 6
HYBRID APPROACHES

The finite-difference approaches have the advantage that they are easy to comprehend, easy to implement and in the case of [5] run in real-time. The downside is that they can easily become unstable. The implicit methods on the other hand have the advantage that they are guaranteed to remain stable, but may not always conserve mass or run in real-time. Some have tried to combine the best of both approaches in an attempt to overcome each of their limitations.

6.1 Foster and Fedkiw

The approach used by Foster and Fedkiw in [11] is a combination of the semi-Lagrangian approach used by Stam [26] that is discussed in section 5.1 and the finite-difference approach used by Foster and Metaxas [12] and discussed in section 4.2. The discretization technique follows the Marker-and-Cell approach used by Foster and Metaxas. However, their biggest contribution is the method used to track the free-surface between the water and the air. Additionally, they describe a simple mechanism to handle objects that move through the liquid.

6.1.1 Tracking the Free Surface

Foster and Fedkiw use a combination of marker particles and a level set method to track the free surface of the water. Both the particles and the level set are updated during each iteration. The values are then compared and combined to yield the final surface for each frame of the simulation.

Marker Particles

The marker particles are assumed to be massless and are distributed throughout the entire fluid volume during initialization. During the simulation, particles are introduced at inflow boundaries. During each iteration the position of each particle is updated using simple kinematics.

$$\mathbf{x}_p = \mathbf{x}_p + \mathbf{u}_{x_p} * \Delta t \tag{6.1.1}$$

where $\mathbf{x}_p$ is the position of the particle and $\mathbf{u}_{x_p}$ is the velocity of the fluid at $\mathbf{x}_p$. The value of $\mathbf{u}_{x_p}$ is obtained using trilinear filtering. While this approach is computationally efficient and easy to implement, it is unclear how the particles should be connected in order to retrieve the surface.

An alternative is to use the particles to locate an isocontour of an implicit function. The implicit function is defined on a high-resolution grid that is placed inside of the Navier-Stokes grid. Each particle represents the center of an implicitly defined surface with radius r,

$$\phi_p(\mathbf{x}) = \sqrt{(x_i - x_{pi})^2 + (x_j - x_{pj})^2 + (x_k - x_{pk})^2} - r \tag{6.1.2}$$

where $\mathbf{x_p}$ is the position of the particle. The surface of the particle is defined as the points where $\phi_p(\mathbf{x}) = 0$. An implicit function $\phi(\mathbf{x})$ can then be defined using all particles by taking the value of $\phi_p(\mathbf{x})$ from the particle closest to $\mathbf{x}$. If $\phi(\mathbf{x})$ is sampled at each point in the high resolution grid, then the Marching Cubes algorithm can be used to tesselate the isocontour where $\phi(\mathbf{x}) = 0$ (see Figure 6.1).

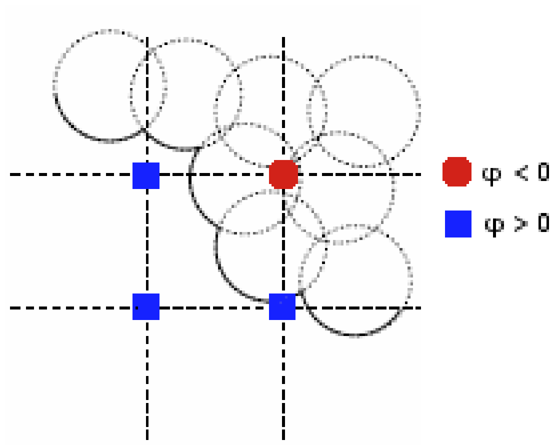


Figure 6.1: The high-resolution grid and the corresponding sign of the function ϕ. Figure directly adapted from [11]

Now that the surface has been found the authors smooth it to yield a less bumpy appearance. The first step in the smoothing process is to normalize the value of $|\phi(\mathbf{x})|$ so that it represents the distance to surface. The sign of ϕ will be set so that it is negative inside the fluid and positive on the outside. The normalization is done using a technique called the Fast Marching Method. The Fast Marching Method will be discussed in the next section.

After $\phi(\mathbf{x})$ has been normalized then the next step in the smoothing process is to smooth out any unnatural folds or corners. The authors use the following smoothing equation

$$\phi_\eta = -S(\phi^{\eta=0})(\|\nabla\phi\| - 1) \tag{6.1.3}$$

to update the values of ϕ close to the $\phi(\mathbf{x}) = 0$ isocontour. Where

$$S(\phi) = \frac{\phi}{\sqrt{\phi^2 + \Delta\tau^2}} \tag{6.1.4}$$

and $\Delta\tau$ is the grid spacing (assuming that $\Delta\tau$ is the same for all directions). The smoothing function is applied for a few relaxation steps in fictitous time η. The smoothed surface is still not perfect. It requires a large number of particles near the $\Phi(\mathbf{x}) = 0$ isocontour. It also requires particles to be present in the entire fluid volume even when they clearly do not contribute to the surface. The solution is to use the particles to create ϕ one time and then track its movement through the same velocity field as that used to move the particles. This is known as a level set. The level set is discussed in section 6.1.1.

Fast Marching Method

Fast Marching Methods are used to solve Eikonal equations that come in the form:

$$\|\nabla u(x, y, z)\| = F(x, y, z)\ in \tag{6.1.5}$$

where $F(x, y, z) > 0$ is a known input and   is a domain in R^2 or R^3. Lastly, $u = g(x)$ where $g(x)$ is a known function on a given curve or surface on . The Fast Marching algorithm is defined as follows.

1. Label initial known values as *Alive*.

2. Label all points that are one grid space away from an *Alive* point as *Close*.

3. Label all remaining points as *Far*.

4. Begin Loop

5. Identify the point in *Close* with the smallest value of u and label this point as *Trial*.

6. Label all neighbors of *Trial* that are not *Alive* as *Close*. If one of these points was in *Far*, then remove it from *Far*.

7. Recompute the values of u at all *Close* neighbors of *Trial*.

8. Add *Trial* to the *Alive* set and remove it from the *Close* set.

9. End Loop

In the context of the fluid simulation, the initial *Alive* set would be all points for which $\phi > 0$.

The level set

Recall the level set equation

$$\frac{\partial \phi}{\partial t} + \mathbf{u} \cdot \nabla \phi = 0 \ . \tag{6.1.6}$$

This is an advection equation similar to the advection term in the Navier-Stokes equation, $(\mathbf{u} \cdot \nabla)\mathbf{u}$. The authors solve this equation using the technique presented in [9]. This is a multi-step process.

1. We need to calculate ϕ_x^+ and ϕ_x^- which represents the partial derivative of ϕ with respect to x in forward and reverse directions respectively.

2. For ϕ_x^- let:

$$v_1 = \tfrac{\phi_{i-2}-\phi_{i-3}}{\Delta x}, \qquad v_2 = \tfrac{\phi_{i-1}-\phi_{i-1}}{\Delta x}$$

$$v_3 = \tfrac{\phi_{i}-\phi_{i-1}}{\Delta x}, \qquad v_4 = \tfrac{\phi_{i+1}-\phi_{i}}{\Delta x}$$

$$v_5 = \tfrac{\phi_{i+2}-\phi_{i+1}}{\Delta x}$$

3. For ϕ_x^+ let:

$$v_1 = \frac{\phi_{i+3}-\phi_{i+2}}{\Delta x}, \qquad v_2 = \frac{\phi_{i+2}-\phi_{i+1}}{\Delta x}$$

$$v_3 = \frac{\phi_{i+1}-\phi_{i}}{\Delta x}, \qquad v_4 = \frac{\phi_{i}-\phi_{i-1}}{\Delta x}$$

$$v_5 = \frac{\phi_{i-1}-\phi_{i-2}}{\Delta x}$$

4. Next we define:

$$S_1 = \frac{13}{12}(v_1 - 2v_2 + v_3)^2 + \frac{1}{4}(v_1 - 4v_2 + 3v_3)^2 \tag{6.1.7}$$
$$S_2 = \frac{13}{12}(v_2 - 2v_3 + v_4)^2 + \frac{1}{4}(v_2 - v_4)^2 \tag{6.1.8}$$
$$S_3 = \frac{13}{12}(v_3 - 2v_4 + v_5)^2 + \frac{1}{4}(3v_3 - 4v_4 + v_5)^2 \tag{6.1.9}$$

5. Next we define:

$$a_1 = \frac{1}{10}\frac{1}{(\epsilon+S_1)^2}, \qquad w_1 = \frac{a_1}{a_1+a_2+a_3}$$
$$a_2 = \frac{6}{10}\frac{1}{(\epsilon+S_2)^2}, \qquad w_2 = \frac{a_2}{a_1+a_2+a_3}$$
$$a_3 = \frac{3}{10}\frac{1}{(\epsilon+S_3)^2}, \qquad w_3 = \frac{a_3}{a_1+a_2+a_3}$$

6. Lastly,

$$(\phi_x^{\pm})_{i0} = w_1\left(\frac{v_1}{3} - \frac{7v_2}{6} + \frac{11v_3}{6}\right) + w_2\left(\frac{-v_2}{6} + \frac{5v_3}{6} + \frac{v_4}{3}\right) + w_3\left(\frac{v_3}{3} - \frac{5v_4}{6} - \frac{v_5}{6}\right)$$

7. When solving the advection equation, if $u_{i0} > 0$ then use ϕ_x^-. If $u_{i0} < 0$, then use ϕ_x^+. If $u_{i0} = 0$, then neither is required.

8. Derivations for y and z are an obvious extension of what was presented here.

Combining the Particles and the level set

After updating both the particles and the level set then the value of the level set is used to determine how to interpret the particles. If a particle is inside and more than a few grid cells away from the surface than it is deleted. This saves computational effort since the particle can not contribute to the surface, then there is no reason to keep it around. Particles are introduced into cells that are close to the $\phi(\mathbf{x}) = 0$ isocontour that contain relatively few particles. This will help to maintain definition in these cells. For each particle that is close to the surface, the curvature of the interface is calculated as:

$$k = \nabla \cdot (\nabla\phi / \|\nabla\phi\|) \tag{6.1.10}$$

Areas with a small value of k have low curvature and are smooth. In these areas particles are ignored and the level set is used to determine the surface of the liquid. In areas of high curvature the particle values are used to modify the value of ϕ obtained from the level set. If the implicit function for a particle at a grid point would give a smaller value of ϕ than the level set, then the smaller value is used to replace the value obtained from the level set. Despite our best efforts, some particles will escape from the liquid. These can be used to indicate mist or spray.

6.1.2 Updating the Velocity Field

Foster and Fedkiw use a combination of the semi-Lagrangian approach that was used by Jos Stam and the finite-difference approach used by Foster and Metaxas. The advection portion of the Navier-Stokes equation is updated using the semi-Lagrangian approach and the diffusion is handled using finite-differences. The semi-Lagrangian approach allows for a large time step, but tracking the fluid surface requires a small time step. The velocity field is updated using a large time step and then the surface is repeatedly updated using a series of small time steps that sum to the value of the large time step. Conservation of mass is enforced using a technique very similar to that used by Jos Stam in [26] except that a Preconditioned Conjugate Gradient method is used rather than a Gauss-Seidel Overrelaxation Method.

6.1.3 Moving Objects

Foster and Fedkiw present a fairly straightforward method for the liquid to respond to moving objects.

1. At the beginning of the frame, set the velocity of any cell within an object equal to the velocity of that object.

2. Update the velocity field as usual. Make no special considerations for cells with objects.

3. The velocities of every cell that intersects a surface has its velocities modified so that the component in the direction of the surface normal is equal to zero. That is $\mathbf{u} \cdot \mathbf{n_s} = 0$.

4. The velocity at cells within the object are set equal to the velocity of the object.

5. When performing the mass conservation step, the velocity is held fixed at any cell that intersects an object.

Although straightforward, the method does have some drawbacks. First, it can only accomodate one polygon per cell. This can be overcome by the averaging the normals for all faces within a cell. Second, the method only models the liquid's response to the object and not the forces that the liquid applies to the object.

6.1.4 Results

All results were obtained on a 500MHz PentiumII and the times indicated include only simulation time, not rendering time. A grid of 250x75x90 cells took about 7 minutes per frame. A grid of 150x75x90 cells took about 4 minutes per frame. A grid of 150x200x150 cells took about 3 minutes per frame.

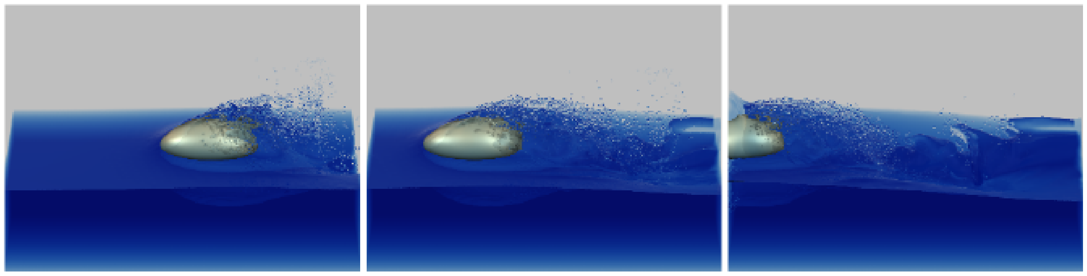


Figure 6.2: An example of an animation created using the technique discussed in section 6.1. The environment consisted of 150x75x90 cells and simulation took approximately four minutes per frame. Image taken directly from [11].

Chapter 7

COLUMN-BASED APPROACHES

The model developed by Holmberg and Wünsche in [14] abandons the Navier-Stokes equations all together. Theirs is a column-based approach that is based on the science of hydrostatics.

7.1 Holmberg and Wünsche

The model proposed by Holmberg and Wünsche in [14] to simulate water can be broken down into two pieces: the volume model and the spray model. The volume model is used to describe the main volume of water while the spray model is used to describe water that has broken free from the main volume.

7.1.1 Volume Model

The simulation area is divided into a 2D grid of columns with equally sized squares as their base. The height of each column is initialized to a user-defined value and represents the depth of the water. Each column is then divided into cells. Virtual pipes are then connected between each of the cells. Figure 7.1 shows a 2D cross-section of the initial set up.

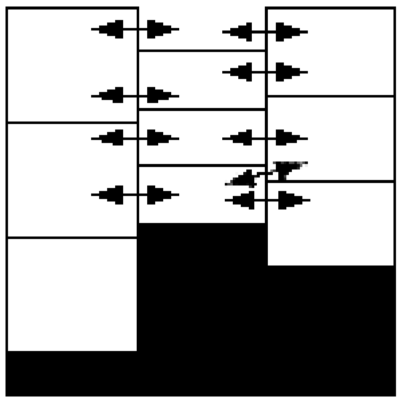


Figure 7.1: A 2D cross-section of the initial column set up used in [14]. Here arrows represent virutal pipes and the solid colored blocks represent terrain.

The amount of liquid that flows through each pipe is dependent on the pressure differential along the pipe. The pressure, Q, of each column is derived from Bernoulli's equation [13] and can be written as

$$Q = h\rho g + \frac{1}{2}\rho\mathbf{v}^2 + (p_0 + E) \tag{7.1.1}$$

where Q is the total pressure, h is the height of the column, g is the acceleration due to gravity, ρ is the fluid density, $\mathbf{v}$ is the velocity of the flow, p_0 is the air pressure and E is the pressure from external forces. The rate of flow through the pipe can then be determined based on Poiseuille's equation [13]

$$\mathbf{v} = f\mathbf{v}_0 + \Delta t\left(\frac{Q_{head} - Q_{tail}}{\rho l}\right) \tag{7.1.2}$$

where l is the pipe length (or grid size), f is the friction coefficient and $\mathbf{v}_0$ is the velocity from the previous time step. Once the flow velocity is calculated, the volume of water passing through the pipe is simply

$$V = \Delta t\mathbf{v}C \tag{7.1.3}$$

where C is the cross sectional area of the pipe defined as

$$\begin{aligned} C &= (h_{top} - h_{bottom}) * d \qquad (7.1.4)\\ h_{top} &= min(a_1, a_2)\\ h_{bottom} &= max(b_1, b_2) \end{aligned}$$

where d is the grid spacing, a_i is the top of two cells and b_i is the bottom of two cells. This can be thought of as the overlap of two cells. This area is shown in red in Figure 7.2. Since the fluid is assumed to be incompressible then the volume flowing into one column must be equal to the volume flowing out the column on the other end of the pipe. In the cases where a column may be left with a negative volume, then the flow quantities need to be adjusted to avoid this situation.

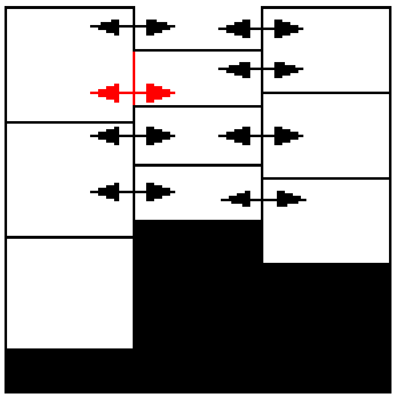


Figure 7.2: The area in red indicates the area calculated by equation (7.1.4).

7.1.2 Spray Model

Generating Particles

The spray model is intended to simulate water that has broken free from the main volume of water. In previous work, such as [6], spray was introduced when the vertical velocity of a column exceeded a pre-determined threshold. This approach has little to no physical justification. Instead, Holmberg and Wünsche use research from the study of waves. It has been shown that a wave becomes unstable when the wave height is 0.78 of the water depth. This same fact is used to determine when particles should be released from the main volume.

Once the need for particles has been identified then the amount of volume that the particles represent needs to be determined. The authors use an adaptation of the Kindsvater-Carter equation [20] which is used to measure the flow rate through a weir. A weir is a device that is placed into a channel or river and is used to measure the flow of water [10]. The flow rate of water through a weir is defined as

$$\zeta = \frac{2}{3} B H^{\frac{3}{2}} \sqrt{2\mathbf{g}} \tag{7.1.5}$$

where B is the base length, H is the height of the water above the base and $\mathbf{g}$ is the force due to gravity. Since ζ is the rate of flow then the total volume represented is given by $V = \zeta \Delta t$. The volume of each individual particle is determined prior to simulation. Therefore, the total number of particles to be created is given by

$$n_{particles} = V / v_{particle} \tag{7.1.6}$$

where $v_{particle}$ is the volume of each individual particle. One additional particle may be created to account for any additional volume left out in the above equation.

Now that the number of particles has been determined then their initial velocities needs to be determined. The authors state

> *To do this the difference in total height between the column for which particles are being generated and the column behind is used in the classic formula*
>
> $$v^2 = u^2 + 2as$$
>
> *Where v is the current velocity, u is the initial velocity, a the acceleration, in this case gravity, and s the distance covered.*

However, it is unclear to which column they are refering to by "... and the column behind". Secondly, they do not provide a reference or derivation for their "classic" formula.

Particle Movement and Reabsorption

Once particles are created they are moved using simple kinematics. The only force acting on particles is gravity.

When a particle hits a column of water, then it exerts a force on the water. The force can be described as

$$\mathbf{F} = v\mu + V\rho\mathbf{g} \tag{7.1.7}$$

Where v is the velocity of the particle, μ is the viscosity, V is the volume of water displaced, ρ is the density of the fluid and $\mathbf{g}$ is gravity. Pressure is defined as $P = \mathbf{F}/Area$. Therefore, it is easy to calcuate the pressure increase at the column where the impact occurred.

7.1.3 Results

On a 1.4GHz PC an un-optimized simulation using 13,000 columns, 4 cells per column and about 3700 particles was able to run at about 15 seconds per frame.

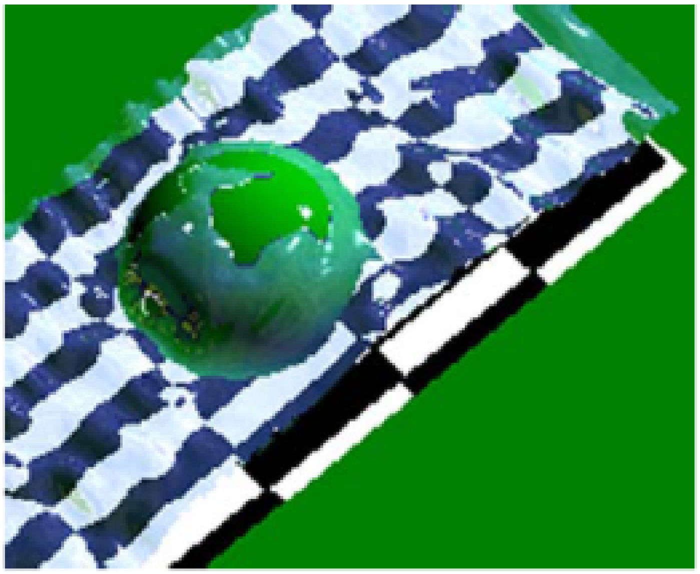


Figure 7.3: Frame from animation using the model described in [14]. Image taken directly from [14].

Chapter 8

SMOOTHED PARTICLE HYDRODYNAMICS

8.1 Introduction

The previous approaches that have been discussed all fall under the category of Eulerian approaches. They have utilized a fixed grid located in space and evaluated the properties of the fluid as it moves past each grid point. An alternative approach is to describe the motion of the fluid using particles that are moving with the flow. This is called a Lagrangian approach. Smoothed particle hydrodynamics (SPH) works in this fashion.

The SPH approach estimates the properties of the fluid at any given point by a weighted average of the property over the nearby fluid volume [21]. Mathematically this can expressed as

$$A(\vec{r}) = \int_0^\tau A(\vec{s})W(|\vec{s}-\vec{r}|)d^3s \tag{8.1.1}$$

where A is some property of the fluid and W is the weighting function. Rather than computing the given integral, the SPH approach approximates it by using a finite set of particles. Thus the above equation becomes

$$A(\vec{r}) \approx \sum_j^N A_j W(\mathbf{r}-\mathbf{r}_j, h)\Delta V_i \tag{8.1.2}$$

Here j iterates over all particles and h is the "core radius" of W. The core radius of the smoothing kernel can be thought of as the maximum distance from j for which $W \neq 0$. Using the fact that $\Delta V_i = \frac{m_i}{\rho_i}$ then equation (8.1.2) becomes

$$A_S(\mathbf{r}) = \sum_j^N m_j \frac{A_j}{\rho_j} W(\mathbf{r}-\mathbf{r}_j, h) \tag{8.1.3}$$

where ρ_j is the density of particle j, and m_j is the mass of particle j. This equation is known as the general SPH equation.

Every particle in the SPH simulation carries mass, position and velocity. Since density is not one of the quantities carried by each particle it is obtained using the SPH equation by setting $A = \rho$.

$$\begin{aligned}\rho_S(\mathbf{r}) &= \sum_j m_j \frac{\rho_j}{\rho_j} W(\mathbf{r} - \mathbf{r}_j, h) \\ &= \sum_j m_j W(\mathbf{r} - \mathbf{r}_j, h)\end{aligned}$$

Some terms of the Navier-Stokes equations rely on the gradient and Lapalacian of field values. Using SPH the gradient and Laplacian of quantity A is defined as the gradient and Laplacian of W

$$\nabla A_S(\mathbf{r}) = \sum_j m_j \frac{A_j}{\rho_j} \nabla W(\mathbf{r} - \mathbf{r}_j, h) \tag{8.1.4}$$

$$\nabla^2 A_S(\mathbf{r}) = \sum_j m_j \frac{A_j}{\rho_j} \nabla^2 W(\mathbf{r} - \mathbf{r}_j, h) \tag{8.1.5}$$

To show this we will follow the derivation given in [17]. In a system with two particles, i and j, the partial derivative of (8.1.3) with respect to x is

$$\frac{\partial}{\partial x} A_S(\mathbf{r_i}) = \frac{\partial}{\partial x} \left(\sum_j m_j \frac{A_j}{\rho_j} W(\mathbf{r_i} - \mathbf{r}_j, h) \right) \tag{8.1.6}$$

Using the product rule of differentiation we get

$$\begin{aligned}\frac{\partial}{\partial x} \left(A_j \frac{m_j}{\rho_j} W(\mathbf{r}_i - \mathbf{r}_j, h) \right) &= \frac{\partial}{\partial x} \left(A_j \frac{m_j}{\rho_j} \right) W(\mathbf{r}_i - \mathbf{r}_j, h) + A_j \frac{m_j}{\rho_j} \frac{\partial}{\partial x} W(\mathbf{r}_i - \mathbf{r}_j, h) && (8.1.7) \\ &= 0 \cdot W(\mathbf{r}_i - \mathbf{r}_j, h) + A_j \frac{m_j}{\rho_j} \frac{\partial}{\partial x} W(\mathbf{r}_i - \mathbf{r}_j, h) && (8.1.8) \\ &= A_j \frac{m_j}{\rho_j} \frac{\partial}{\partial x} W(\mathbf{r}_i - \mathbf{r}_j, h) && (8.1.9)\end{aligned}$$

Assuming that the value of A is constant throughout particle j then $\frac{\partial}{\partial x}\left(A_j \frac{m_j}{\rho_j}\right) = 0$. The parital derivative with respect to y and z follow a similar derivation thus allowing the gradient of A to be written as equation (8.1.6). A similar derivation can be taken for the Laplacian of A.

8.1.1 Lagrangian Approach to Solving the Navier-Stokes Equations

Recall equation (3.0.2) is the continuity equation and states that for an incompressible flow, mass must be conserved. Since each particle in the system represents a distinct mass quantity then mass is guaranteed to always be conserved. Therefore equation (3.0.2) can be disregarded completely.

When discussing the Lagrangian approach to solving the Navier-Stokes conservation of momentum equation it is convenient to rewrite equation (3.0.1) as [23]

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u}\right) = -\nabla p + \rho\mathbf{g} + \mu\nabla^2\mathbf{u}\ . \tag{8.1.10}$$

Once again, the choice of particles as fluid representation can simplify equation (8.1.10). The advection term $(\mathbf{u}\cdot\nabla)\mathbf{u}$ is used to describe how the velocity flows with itself. Since the particles both define the velocity and move with it, then the advection term has already been satisfied and does not need to be explicity calculated. This reduces equation (8.1.10) to

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t}\right) = -\nabla p + \rho\mathbf{g} + \mu\nabla^2\mathbf{u}\ . \tag{8.1.11}$$

The right hand side of equation (8.1.11) represents the sum of the forces acting on each particle, $\mathbf{F}_i$. Using Newton's equation which states that $\mathbf{F} = m\mathbf{a}$ we can derive the acceleration for each particle, i, as

$$\mathbf{a}_i = \frac{\mathbf{F}_i}{\rho_i}\ . \tag{8.1.12}$$

Once the accleration of particle i has been determined, then its position and velocity can be updated using any numerical integration scheme that one prefers.

8.1.2 Calculating Forces

Each force on the right-hand side of equation (8.1.11) can be calculated using equation (8.1.3). However, in some cases the results will not be very accurate and some modifications can be made to overcome these shortcomings.

Pressure

The force due to pressure, $\mathbf{f}_i^{\text{pressure}}$ is represented in the Navier-Stokes equation as $-\nabla p$. Plugging pressure into equation (8.1.4) gives

$$\mathbf{f}_i^{\text{pressure}} = \sum_j m_j \frac{p_j}{\rho_j} \nabla W(\mathbf{r}_i - \mathbf{r}_j, h) \tag{8.1.13}$$

This equation presents two issues. The first is that in order to calculate the pressure force at position $\mathbf{r}_i$ we need to know the value of the pressure at position $\mathbf{r}_j$. The second is that the resulting pressure force is likely not going to be symmetric. The first issue is resolved using the ideal gas law [17]

$$pV = nRT \tag{8.1.14}$$

where p is pressure, $V = \frac{1}{\rho}$ is the volume per unit mass, n is the number of molecules in a mol, R is the universal gas constant and T is the temperature. In the case of an isothermal fluid with a constant mass then the right hand side can be reduced to a single scalar k. Thus, the pressure can be written as

$$\begin{aligned} pV &= k \\ p\frac{1}{\rho} &= k \\ p &= k\rho \end{aligned} \tag{8.1.15}$$

To understand the second issue with equation (8.1.13) consider a system with only two particles, a and b. Particle a will only use the pressure of particle b to compute its pressure force and vice versa. The only way that the force can be symmetric is if particles a and b had the same pressure. In general this will not be the case. In [23] M**ü**ller et. al resolved this issue by simply using the arithmetic mean of the two pressures. Thus the resulting equation for the force due to pressure is defined as

$$\mathbf{f}_i^{\text{pressure}} = \sum_j m_j \frac{p_j + p_i}{2\rho_j} \nabla W(\mathbf{r}_i - \mathbf{r}_j, h) \ . \tag{8.1.16}$$

There exists one last issue with calculating the pressure force using equations (8.1.15) and

(8.1.16). The resulting force will always result in repulsive forces [17]. While this may be true for a gas which is always expanding, it is not valid for a liquid that should exhibit internal cohesion and have a constant mass-density when at rest. The solution discussed in [17] is to introduce an additional rest pressure, p_0, such that

$$\begin{aligned}
(p + p_0)V &= k \\
(p + k\rho_0)V &= k \\
(p + k\rho_0)\frac{1}{\rho} &= k \\
p + k\rho_0 &= k\rho \\
p &= k(\rho - \rho_0) \qquad (8.1.17)
\end{aligned}$$

Using the pressure given by equation (8.1.17) in equation (8.1.16) will result in a pressure force that will act to keep the fluid at its rest density.

Viscosity

Applying the SPH equation to viscosity yields

$$\mathbf{f}_i^{\text{viscosity}} = \mu\nabla^2\mathbf{u}(\mathbf{r}_i) = \mu\sum_j m_j\frac{\mathbf{u}_j}{\rho_j}\nabla^2 W(\mathbf{r}_i - \mathbf{r}_j, h) \qquad (8.1.18)$$

This again yields an asymmetric force. The solution proposed in [23] is a keen observation that the viscosity of the fluid only depends on the relative velocity between particles. Thus the equation becomes

$$\mathbf{f}_i^{\text{viscosity}} = \mu\sum_j m_j\frac{\mathbf{u}_j - \mathbf{u}_i}{\rho_j}\nabla^2 W(\mathbf{r}_i - \mathbf{r}_j, h) \qquad (8.1.19)$$

8.1.3 Determining Surface Location

The free surface between the fluid and the air is obtained using an additional field that the literature calls the *color field*. The color field, c_j, is defined to be 1 at particle locations and 0 everywhere else. Plugging the color field into the general SPH equation (8.1.3) yields

$$\begin{aligned} c_S(\mathbf{r}) &= \sum_j m_j \frac{c_j}{\rho_j} W(\mathbf{r} - \mathbf{r}_j, h) \\ &= \sum_j m_j \frac{1}{\rho_j} W(\mathbf{r} - \mathbf{r}_j, h) \quad \text{Using the fact that } c_j \text{ is always 1} \end{aligned}$$

The normal to the surface, $\mathbf{n}$ is defined as the gradient of the color field

$$\mathbf{n} = \nabla c_S(\mathbf{r}) \tag{8.1.20}$$

The final location of the surface is defined at points where $|\mathbf{n}| > l$ where l is a pre-determined threshold value. Once the surface has been determined then Müller et. al suggest in [23] that it can be rendered using either point splatting or the Marching Cubes algorithm. They note that Marching Cubes produces a better nicer result but takes much longer.

8.2 Müller, Charypar and Gross

The major contributions made by Müller et. al in [23] are smoothing kernels for various portions of the SPH simulation, the description of an addtional force due to surface tension and the description of results they have obtained using the SPH approach.

8.2.1 Smoothing Kernels

In [23], Müller et. al use the following smoothing kernel for all but two calculations

$$W_{default}(\mathbf{r}, h) = \frac{315}{64\pi h^9} \begin{cases} (h^2 - |\mathbf{r}|^2)^3, & \text{if } 0 \le |\mathbf{r}| \le h \\ 0, & \text{otherwise} \end{cases} \tag{8.2.1}$$

The value of this kernel, its gradient and Laplacian is shown in Figure 8.1. One nice feature of this kernel is that it does not involve computing the magnitude of $\mathbf{r}$ which avoids a computationally expensive square root calculation.

The kernel given by equation 8.2.1 can not be used to calculate the force due to pressure. The problem is that the gradient goes to zero as the two particles get close together. This would erase any repulsive forces between the two particles. In [23], Müller et. al solve this problem using the so called "spiky" kernel given by

$$W_{spiky}(\mathbf{r}, h) = \frac{15}{\pi h^6} \begin{cases} (h - |\mathbf{r}|)^3, & \text{if } 0 \le |\mathbf{r}| \le h \\ 0, & \text{otherwise} \end{cases} \tag{8.2.2}$$

The value of this kernel, its gradient and Laplacian is shown in Figure 8.1.

The last kernel introduced by Müller et. al in [23] is used to compute the force due to viscosity. The kernel used to compute the viscosity force must have a Laplacian that is always positive. To understand this recall that viscosity is a force that is created by friction and decreases a fluids kinetic energy by converting it into heat. If the smoothing kernel were ever allowed to be negative then it could end up increasing the relative velocity between particles. The kernel proposed in [23] is defined as

$$W_{viscosity}(\mathbf{r}, h) = \frac{15}{2\pi h^3} \begin{cases} -\frac{|r|^3}{2h^3} + \frac{|r|^2}{h^2} + \frac{h}{2|r|} - 1, & \text{if } 0 \le |\mathbf{r}| \le h \\ 0, & \text{otherwise} \end{cases} \tag{8.2.3}$$

The value of this kernel, its gradient and Laplacian is shown in Figure 8.1.

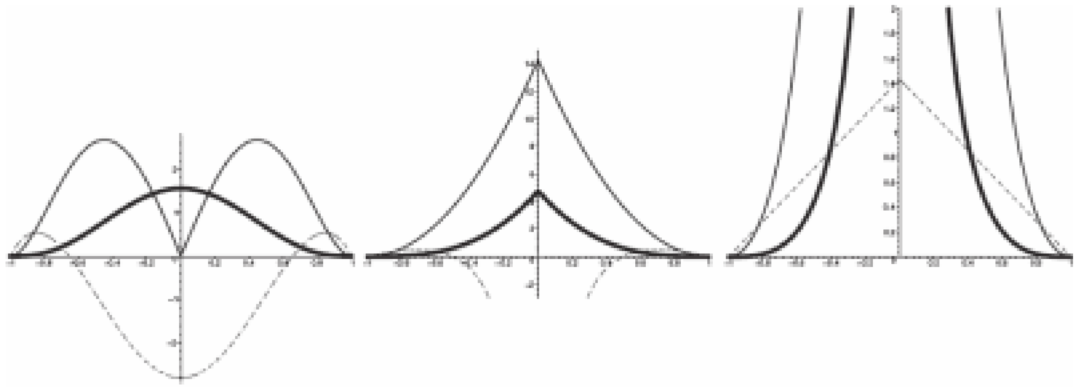


Figure 8.1: The three kernels , $W_{default}$, W_{spiky} and $W_{viscosity}$ (from left to right) proposed by Müller et. al in [23]. The thick lines show the kernels, the thin lines their gradients and the dashed lines the Laplacian. Image taken directly from [23].

8.2.2 Surface Tension

In [23], Müller et. al introduce an additional force not found in the Navier-Stokes equation. This is the force due to surface tension. Fluid molecules are constantly subject to attractive forces from nearby molecules. Inside the fluid all of these attraction forces balance each other. However, on the surface the force is unbalanced.

The surface tension force is directly related to the curvature of the surface. The curvature is defined as the divergence of the normal, **n**, as defined in section 8.1.3.

$$\kappa = \nabla \cdot \frac{\mathbf{n}}{|\mathbf{n}|} = -\frac{\nabla^2 c_S(\mathbf{r})}{|\mathbf{n}|} \tag{8.2.4}$$

where c_S is the color field as discussed in section 8.1.3. The final equation given for surface tension in [23] is

$$\mathbf{f}^{\text{surface}} = \sigma\kappa\mathbf{n} = -\sigma\nabla^2 c_S \frac{\mathbf{n}}{|\mathbf{n}|} \tag{8.2.5}$$

where σ is a tension coefficient that depends on the two interacting fluids.

8.2.3 Results

Results obtained on a 1.8GHz Pentium 4 with a GeForce 4 graphics card. Times include simulation and rendering. A system with 2200 particles rendered using point splatting was able to achieve 20 fps. Another system that included 1300 particles and allowed for interaction with user-supplied forces was able to achieve 25 fps. A third system rendered using the Marching Cubes algorithm and containing 3000 particles was able to achieve 5 fps.

8.3 Kipfer and Westerman

In [18] the authors Kipfer and Westerman strive to simulate rivers using SPH and sparse particle sets. One of the major contributions made by Kipfer and Westerman in [18] is a spatial data structure to easily locate the nearest neighbor of each particle when evaluating the SPH equations and when determining collision between particles. Additionally, they present a technique to extract the surface of the water for rendering. The technique can be implemented either on the CPU or entirely on the GPU. Lastly, they discuss joining the two techniques together.

8.3.1 Spatial Data Structure

In [18], Kipfer and Westerman propose the use of a staggered grid data structure that can be used to determine potential collisions between particles. Additionally, the data structure can be used to determine a particle's nearest neighbors, useful when evaluating the SPH equation (8.1.3). The computational domain is divided into a grid of three-dimensional cells with each side of length $2r$ where r is the maximum radius of a particle. This structure is similar to the leaf-level of an octree. Each particle's position is then used to determine the indices, (i_x, i_y, i_z) of the cell in which the particle currently resides. The indices are combined into a single 64-bit identifier, $(id = 0|i_y|i_z|i_x)$. Note that i_x has been placed at the end. This will be explored shortly. The particles are then placed into a list and sorted by their identifier in descending order. The list is then linearly traversed starting from the left-most entry and progressing to the right checking for collision with each particle until a particle identifier is found such that $id_{particle} \leq (id_{self} - 2)$. Since i_x was placed as the least-significant bits of the identifier then this collision detection is only valid for the x-dimension. Therefore, two additional lists need to be created and sorted, one for the y-dimension and one for the z-dimension. Once a collision is resolved, then the leading bit of the particles' identifier is set to 1 so that the collision is processed by subsequent list traversals. As shown in figure 8.2, Kipfer and Westerman demonstrated significant speed improvements using this data structure compared to a more traditional octree.

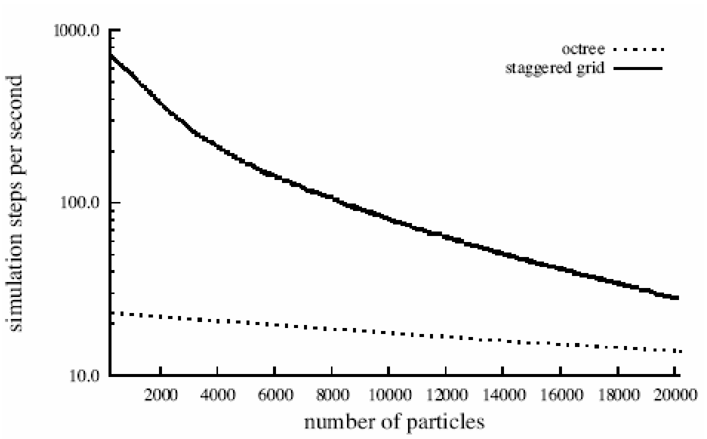


Figure 8.2: A graph demonstrating the speed increase of the staggered grid data structure over a more traditional octree implementation. Image taken directly from [18]

8.3.2 Surface Extraction - The Carpet Method

In [18], Kipfer and Westerman propose a technique called the “carpet method” to extract and animate the surface of water. They provide two forms of implementation, one that runs on the CPU and a second that runs on the GPU. In a scene with 20,000 particles, they reported a speed up of almost four times when using the GPU method as opposed to the CPU method. The goal of the surface extraction method is to quickly build a closed surface that encompasses all particles.

CPU-Based Surface Extraction

A quadtree is laid over the computational domain. Each leaf-node of the quadtree is initialized with the maximum height value all of particles inside the leaf, the minimum height value of the terrain inside the leaf and a velocity value of zero. Both the he maximum particle height value and the minimum terrain height value are then propogated up the tree. The carpet can then be efficiently rendered by traversing the tree and examining the two height values. If at any point the maximum particle height is less than the terrain height then the recursion can stop. If a leaf node is reached then the corresponding quad is rendered. After the quadtree is traveresed and rendered then each quad is accelerated downward by gravity. If the particle that was supplying the maximum height value has left the quad, then the quad will slowly fade down below the terrain. If not, then the quad will be rendered again during the next frame.

GPU-Based Surface Extraction

The GPU-Based method for constructing the carpet utilizes two vertex buffers. The vertices of the first buffer are accelerated downward according to gravity as they are drawn into the second buffer. The height is stored as the z-value. This step is analogous to the gravity acceleration performed in the CPU construction. The particles are then splatted into the new carpet vertex buffer with z-testing turned on. This will have the effect of raising the carpet vertices at points where the particles are above the lowered carpet. Lastly, the newly created buffer is drawn as an array of triangle strips and the original two vertex buffers are swapped for the next iteration. Splatting, sometimes called surface splatting or point splatting, is a technique that is used to render an image of an object whose color and normal are defined at specific unconnected points.

A complete discussion of point splatting is beyond the scope of this paper. The interested reader is referred to [32].

8.3.3 Joining the Spatial Data Structure and Carpet Method

Using the CPU-based method to render the carpet as described in section 8.3.2 would require a very high resolution quadtree. This would require a great deal of memory and traversal times would become unacceptable. Instead, the quadtree is defined to contain a set of the spatial bins used for the data structure described in section 8.3.1. When rendering the carpet using the CPU method, the topology of each quad will depend on which spatial bins contain particles. Additionally, if a particle is in a bin far from its neighbors then it could be used to represent spray and perhaps be rendered with a different texture.

When using the GPU-based carpet approach one is not able to arbitrarily choose rendering primitives. Additionally, there is no need to join the spatial data structure to the GPU-based carpet method as the whole carpet must be rendered everytime regardless.

8.3.4 Results

Results were obtained on a 2.2 GHz AMD Athlon64 with an nVIDIA GeForce 6800 GT graphics card with 256 MB of video memory. A simulation containing 3000 particles was able to run at 68 fps. Another simulation using 8000 particles was able to achieve 26 fps and a third using 20,000 particles was able to run at 12.7 fps using the CPU-based carpet method. All times include simulation and rendering.

Chapter 9

SUMMARY

Many different approaches to simulating the motion of fluids and particularly water have been presented. This section provides a brief comparison of the different techniques based on their realism, speed, storage requirements and stability.

9.1 Realism

The heightfield approach discussed in section 4.1 solves the Navier-Stokes equations in 2D and then scales the pressure at each grid point to determine the height of the surface above the terrain. This is clearly just an approximation technique that neglects the activity of the fluid at any point other than the surface. Additionally, since the height above the terrain is defined as a single scalar value, this approach does not allow for overturning waves. The column-based approach presented in section 7 suffers a similar fate. Additionally, the column-based approach is based on the science of hydrostatics and therefore ignores many of the important properties of a fluid in motion.

The grid-based approach discussed in section 4.2 simulates the full 3D Navier-Stokes equations and presents several different possible techniques to track the surface of the fluid. One of the surface tracking techniques involves tracking the position of massless marker particles as they move through the fluid. The surface tracking technique developed by Foster and Fedkiw and discussed in section 6.1.1 can be seen as an extension of this technique. Instead of relying solely on the particles, the authors additionally use a level set technique to extract the surface and then use the particles to modify the level set values. This presents an extremely realistic effect (see Figure 6.2).

The stable method discussed in section 5.1 also solves the full 3D Navier-Stokes equations but the semi-Lagrangian technique used in the advection step tends to lead to unacceptable dissapation of mass. This problem is corrected by Song et. al and is discussed in section 5.2. The level set technique used to extract the surface combined with their approach to simulate air bubbles in the water also presents extremely realistic results, see figure 5.6.

Lastly, the smoothed particle hydrodynaimcs approach discussed in section 8 uses a Lagrangian approach to solve the full 3D Navier-Stokes equations. The accuracy of the approach depends largely on the smoothing kernels that are chosen. The technique can be shown to have second order accuracy, [19], if the kernels meet the three following conditions. The first condition is the *normalization condition* which states

$$\int W(\mathbf{x}-\mathbf{x}', h)d\mathbf{x} = 1 \tag{9.1.1}$$

The second condition is called the *Delta function property* and it states

$$\lim_{h\to 0} W(\mathbf{x}-\mathbf{x}', h) = \delta(\mathbf{x}-\mathbf{x}') \tag{9.1.2}$$

where

$$\delta(\mathbf{x}-\mathbf{x}') = \begin{cases} 1 & \text{if } \mathbf{x}=\mathbf{x}' \\ 0 & \text{if } \mathbf{x}\neq\mathbf{x}' \end{cases} \tag{9.1.3}$$

The third and final condition is known as the *compact condition* and states

$$W(\mathbf{x}-\mathbf{x}', h) = 0 \quad \text{when} \quad |\mathbf{x}-\mathbf{x}'| > \kappa h \tag{9.1.4}$$

where κ is a constant that defines the effective area of the smoothing function at $\mathbf{x}$.

9.2 Speed

The Stable Fluids method developed by Stam and discussed in section 5.1 is the only fully 3D Eulerian approach discussed in this paper that is capable of producing real-time results. The problem is that the semi-Lagrangian approach used to calculate the advection portion of the Navier-Stokes equations leads to an unacceptable amount of mass dissapation. The technique presented by Song et. and discussed in section 5.2 overcomes the mass dissapation but the extra computational complexity prevents the simulation from producing 3D results running in real-time.

The 2D heightfield approach developed by Chen and Lobo (section 4.1) is able to achieve real-time results but at the expense of neglecting the motion of the fluid any where other than at the surface. The particle-based approach presented by Müller et. al and discussed in section 8.2 is a Lagrangian approach that manages to simulate the full 3D Navier-Stokes equations and run in real-time. The downside to the particle method is that it is generally prohibitive to use for large bodies of water because it requires a very large number of particles.

The model developed by Foster and Fedkiw that is presented in section 6.1 offers speeds around 3 minutes per frame. While clearly not capable of real-time results, for an off-line renderer the results are very promising.

9.3 Storage Requirements

The Marker-and-Cell discretization technique used by Chen and Lobo and discussed in section 4.1 stores the pressure values of the grid at the cell centers and the velocities on the cell faces. For a square grid of size w x h there are $\{(h+1)(w)+(h)(w+1)\}$ unique locations where the velocity is stored. The pressure is defined at $w*h$ locations and each requires the storage of a single scalar value.

For a square grid of w x h x d, the 3D Marker-and-Cell method used by Foster and Metaxas (section 4.2) and Foster and Fedkiw (section 6.1) stores the velocity at $\{(d+1)[(h+1)(w)+(h)(w+1)]+(h+1)(w-1)(d-1)\}$ unique locations. The pressure is defined at $w*h*d$ locations and each requires the storage of a single scalar value.

Using a level set technique as proposed by Song et. al [25] or Foster and Fedkiw [11] requires the additional storage space for a single scalar representing the signed distance to the surface at each grid point. The use of marker particles as discussed in sections 4.2.4 and 6.1 requires storing an extra 3D vector represnting position for each particle in the simulation.

The particle-based approach discussed in section 8 requires storing the position, mass and velocity for each particle in the simulation. Since velocity and position are both 3-tuples, then each particle requires storage of seven scalar values. Additionally, during the simulation the density at each particle location needs to be calculated and stored for use in future computations. This increases the number of scalars to eight per particle.

The column-based approach discussed in section 7 requires storing the pressure and velocity for each column as well as two pointers for each of the virtual pipes used in the simulation.

9.4 Stability

The only two techniques discussed in this paper that are guaranteed to be stable are the two that are discussed in section 5. These methods are considered unconditionally stable because the simulation is guaranteed to never diverge regardless of how large a timestep is used. The Eulerian methods that use an explicit discretization such as those discussed in sections 4.1, 4.2, 6.1, and 7 are not guaranteed to be stable but stability can be aided by abiding by the Courant-Friedrichs-Levy (CFL) Condition.

The CFL condition states that the time step used in a numerical simulation should be smaller than the amount of time for "something significant to happen." In the case of Eulerian fluid simulation this is typically defined as the amount of time for a discrete fluid element to travel from one grid point to another. In [11], Foster and Fedkiw suggest a CFL condition such that $\Delta t = \Delta\tau/|\mathbf{u}|$ where Δt is the time step, $\Delta\tau$ is the grid spacing and $\mathbf{u}$ is the velocity.

The stability of the particle-based approach discussed in section 8 is largely dependent on the numerical integration technique used to update velocities and positions based on forces. While simple Euler integration will suffice, stability can be increased using higher-order integration techniques such as RK (Runge-Kutta) 2 or RK-4, [31].

Chapter 10

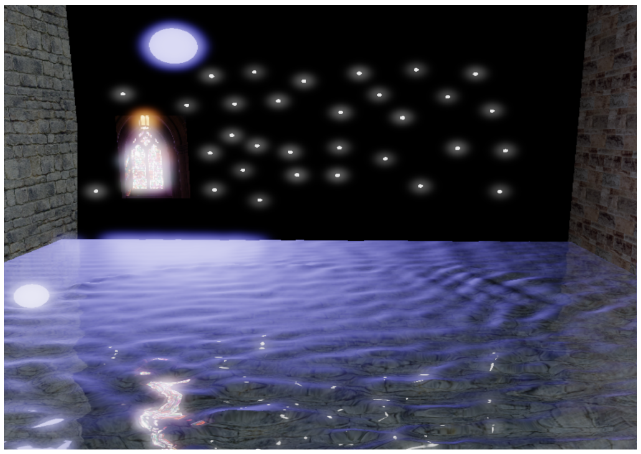
IMPLEMENTATION AND RESULTS

The two following algorithms were implemented: the Chen and Lobo heightfield technique discussed in section 4.1 and the Stable Fluids technique discussed in section 5.1. They were both implemented on the same hardware consisting of a Pentium D processor running at 3.2GHz with 1 gigabyte of RAM and an NVidia GeForce 6800. The simulations were implemented in C++ using Microsoft Visual Studio .NET and run on the Microsoft Windows XP Professional operating system. The DirectX 9.0c API was used for rendering.

10.1 Chen and Lobo

Using a grid of size 80x80, a serial implementation of the Chen and Lobo heightfield technique is able to update the water simulation at an average of rate of 76 times per second. The implementation is able to simulate the motion of the water and render the surface with reflection mapping, refraction mapping and high dynamic range to low dynamic range tonemapping at an average rate of 40 frames per second. A single frame of the simulation is shown below.

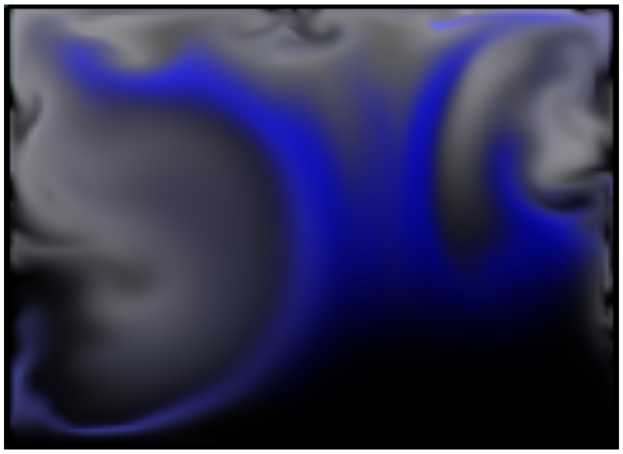
Figure 10.1: A single frame of the Chen and Lobo simulation. The simulation includes reflection and refraction mapping as well as high dynamic range rendering.



10.2 Stable Fluids

While the Chen and Lobo technique was used to simulate water, the Stable Fluids technique was used to simulate smoke. This algorithm was implemented both as a CPU-based simulation and as a GPU-based simulation. Both simulations use a 128x128. The CPU-based implementation is single threaded and is able to update the simulation and render the results at a rate of approximately 25 frames per second. The GPU-based implementation is able to update the simulation and render the results at approximately 50 frames per second. A single frame of the simulation is shown below.

Figure 10.2: A single frame of the Stable Fluids simulation. The simulation allows the user to paint 'virutal smoke' on the screen and then disturb the velocity field and observe the result.



Chapter 11

FUTURE WORK

11.1 Parallel Approach

It has been shown that both algorithms are able to produce to very realistic looking results in real-time. However, Eulerian fluid simulations such as these contain a great deal of inherent parallelism. Typically each grid point can be updated individually and only requires knowledge of its nearest neighbors. This fact combined with the increasing popularity of multi-core processors provides potential for extremely fast simulations. A simple approach to parallelizing the problem would be to divide all of the grid points into n sets, where n is the number of processors available on the system. All of the processors could then be working on their own sets simultaneously.

Another approach to parallelizing the Chen and Lobo technique would be to implement it on the GPU, such as was done with the Stable Fluids method. Given the relative simplicity of the Chen and Lobo technique, it is reasonable to assume that an experienced GPU-developer could implement the algorithm with ease. Additionally, the project may prove extremely worthwhile as a learning exercise for a novice GPU-developer.

11.2 Particle and Heightfield Hybrid Approach

Both the smoothed particle hydrodynamics and the 2D heightfield approaches are able to be used to create real-time simulations. However, each has its own problem. The biggest problem with the heightfield approach is that the scalar-valued height of the surface above the terrain does not allow for overturning water. Additionally, their does not exist a physically justified way to introduce spray. However, it is very efficient when used to simulate large bodies of water where the focus of interest is on the surface. The particle-based approach is very capable of simulating over-turning water and spray is automatically modeled by particles that have strayed from their neighbors. The problem with the particle based approach is that the number of particles required to simulate very large bodies of water can become computationally prohibitive because particles are required throughout the entire fluid domain.

A hybrid approach can take advantage of the strengths of each technique will working to remove their respective weaknesses. The proposed hybird model simulates the main volume of the water using a 2D heightfield approach analogous to that used by Chen and Lobo in [5] and discussed in section 4.1. As the simulation progresses particles can be introduced at “points of interest”. “Points of interest” may include locations such as the boundary between the water and static or dynamic objects in the scene. A process must be developed that allows interaction between the two different simulation techniques. The main considerations involve developing a physically-justified method of advancing the simulation, determining how to convert water from the heightfield into particles and how to merge particles back into the heightfield when necessary.

During every iteration of the simulation the 2D heightfield is updated using the technique discussed in section 4.1.2. After the heightfield has been updated, each of the particles is updated using the smoothed particle hydrodynamics approach discussed in section 8. The particles interact with each other exactly as described in the above section and treat the water simulated by the heightfield as simply a very dense collection of particles.

Imagine the scenario depicted in figure 11.1. The figure shows a cross-section of the heightfield and particles resting on the surface. Recall the generic SPH equation presented in section 8.

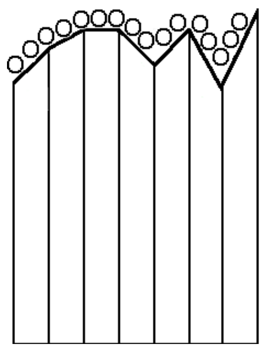


Figure 11.1: A 2D cross-section of an example frame from the proposed technique.

$$A_S(\mathbf{r}) = \sum_j m_j \frac{A_j}{\rho_j} W(\mathbf{r} - \mathbf{r}_j, h) \tag{11.2.1}$$

Here A is any scalar-valued quantity of the fluid. When updating each particle three forces need to be calculated as described in section 8, these are pressure, viscosity and external forces. The inclusion of external forces is trivial and the others will be discussed below.

11.2.1 Columns and Volume

A "column" in the heightfield is defined as three grid points that lie on the vertices of a triangular region such as those shown in Figure 11.2. The height of the column, h_c, is defined as the average pressure at each grid point times the user-defined height scale value, s, plus the user-defined base height of the surface, h. Let the pressure at the three grid points for a given column be defined as p_1, p_2 and p_3. The height of the column can be defined as

$$\begin{aligned} h_c &= h + s(p_c) \quad (11.2.2) \\ p_c &= \frac{p_1 + p_2 + p_3}{3} \quad (11.2.3) \end{aligned}$$

Here the variable h is used to indicate the height of the surface above the terrain when the pressure at each of the grid points is zero.

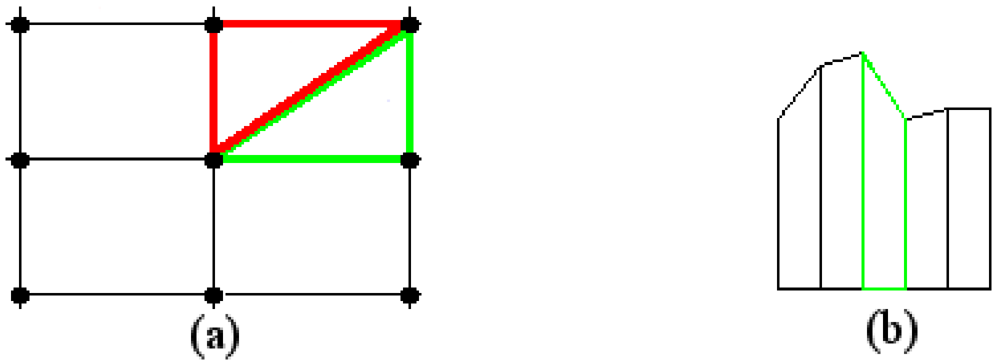


Figure 11.2: (a) A top-down view of the 2D computational grid. The red and green triangles each represent a single triangular column. (b) A 2D side view of the column definition. The region in green defines a single column in the heightfield simulation.

Using the definition of the height of a column given by equation (11.2.2) then the volume in a given column, v_c, is defined as simply the height of the column times the area of its triangular base (the green area shown in Figure 11.2.

$$v_c = \frac{h_c * \Delta x * \Delta y}{2} \quad (11.2.4)$$

11.2.2 Calculating Pressure Force

Recall the SPH equation used for calculating the force due to pressure

$$\mathbf{f}_i^{\text{pressure}} = \sum_j m_j \frac{p_j + p_i}{2\rho_j} \nabla W(\mathbf{r}_i - \mathbf{r}_j, h) \ . \tag{11.2.5}$$

In the hybrid technique, when updating an SPH particle, i, equation (11.2.5) would be used directly when calculating the pressure force due to other nearby SPH particles. However, the particle is also influenced by the the main volume of water being simulated by the heightfield technique. To include these forces the volume of fluid that is contained within the core radius around each particle needs to be considered. This is the red area in Figure 11.3. Assuming that the area in red inside of each column is completely filled with other SPH particles, then the total contribution from the heightfield to the pressure term for the single red particle would be

$$p_S(\mathbf{x}) = \sum_c \int_{V_c} m_c \frac{p_c + p_i}{2\rho_c} \nabla W(\mathbf{r}) \cdot d^3\mathbf{r} \tag{11.2.6}$$

where c iterates over all of the columns in i's core radius and V_c is the volume in column c inside of i's core radius. Since it can assumed that the pseudo-particles inside of the column are packed tight enough for their density to be constant and using the fact that volume equals mass over density, $V = \frac{m}{\rho}$, and since the heightfield approach assumes that the pressure is constant along the entire column, then equation (11.2.6) can be written as

$$p_S(\mathbf{x}) = \sum_c V_c \frac{p_c + p_i}{2} \int_{V_c} \nabla W(\mathbf{r}) \cdot d^3\mathbf{r} \tag{11.2.7}$$

Using the Green-Gauss theorem from calculus, equation (11.2.7) can be rewritten as the following surface integral [4]

$$p_S(\mathbf{x}) = \sum_c V_c \frac{p_c + p_i}{2} \int_{S_c} W(\mathbf{r}) \mathbf{n} da \tag{11.2.8}$$

where $\mathbf{n}$ is the outward pointing surface normal. Depending on the smoothing kernel this integral can either be computed analytically or computed using a numerical integration technique such as Gaussian Quadrature [31]. Alternatively, one could consider the volume in the column to represent one very large particle and the total contribution to the pressure force from all of the columns would simplify to

$$p_S(\mathbf{x}) = \sum_c V_c \frac{p_c + p_i}{2} \nabla W(\mathbf{r}) \tag{11.2.9}$$

where $\mathbf{r}$ is the vector from the SPH particle to the center of the red area in each column as shown in Figure 11.3.

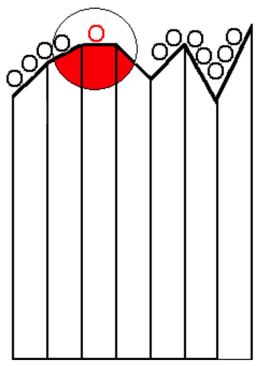


Figure 11.3: A 2D cross section of the proposed technique. Here the red particle is the particle currently under consideration. The circle represents the core radius around the particle. The area colored in red represents the volume of water from the heightfield simulation that needs to be considered when updating the red particle. Note that the volume in each of the three columns needs to be considered seperately and then summed to achieve the final result.

11.2.3 Determining Pressure

The previous section assumes that pressure is defined at all particle locations. Recall from section 8.1.2 that the equation for pressure at each particle is

$$p = k(\rho - \rho_0) \tag{11.2.10}$$

which requires the calculation of the density at the particle location. Additionally recall the equation for density presented in section 8

$$\rho_S(\mathbf{r}) = \sum_j m_j W(\mathbf{r} - \mathbf{r}_j, h)$$

When determining the density of the fluid for a particle, i, the SPH equation above can be used directly when considering other nearby SPH particles. However, just as in the previous section, the particle also needs to consider the main volume of fluid. Referring to figure 11.3, the total contribution of the main fluid volume to particle i is

$$\rho_i = \sum_c m_c \int_{V_c} W(\mathbf{r}, h) \tag{11.2.11}$$

Using equation (9.1.1) and the fact that the density of water is one and thus, $m_c = V_c$, then the above equation simplifies to

$$\rho_i = \sum_c V_c \tag{11.2.12}$$

11.2.4 Viscous Force

Recall the SPH equation for calculating the force due to viscosity

$$\mathbf{f}_i^{\text{viscosity}} = \mu \sum_j m_j \frac{\mathbf{u}_j - \mathbf{u}_i}{\rho_j} \nabla^2 W(\mathbf{r}_i - \mathbf{r}_j, h) \tag{11.2.13}$$

When updating the SPH particles in the proposed technique, equation (11.2.13) can be applied directly to all of the particles in the simulation. However, the SPH particles also need to take into consideration the main volume of water. Using a procedure similar to that described in section 11.2.2, the volume in each column that lies within the particle's core radius contributes to the viscous forces that it experiences. Thus, the total contribution to the viscous force for all columns can be computed as

$$\mathbf{f}_i^{\text{viscosity}} = \sum_c V_c \mu (\mathbf{u}_c - \mathbf{u}_{particle}) \int_{V_c} \nabla^2 W(\mathbf{r}, h) d^3\mathbf{r} \tag{11.2.14}$$

Using the divergence theorem, equation (11.2.14) can be rewritten as

$$\mathbf{f}_i^{\text{viscosity}} = \sum_c V_c \mu (\mathbf{u}_c - \mathbf{u}_{particle}) \int_{S_c} \nabla W(\mathbf{r}, h) da \tag{11.2.15}$$

Depending on the smoothing kernel, W, this integral can either be computed analytically or using a numerical integration technique such as Gaussian Quadrature [31]. Alternatively, one could consider the volume in the column to represent one very large particle and the total contribution to the force due to viscosity from the main volume of water would simplify to

$$\mathbf{f}_i^{\text{viscosity}} = \sum_c V_c \mu (\mathbf{u}_c - \mathbf{u}_{particle}) \nabla^2 W(\mathbf{r}, h) \tag{11.2.16}$$

where $\mathbf{r}$ would be defined as the vector from the SPH particle to the center of the red area in each column as shown in figure 11.3.

11.2.5 Determining Volume Inside of Core Radius

When calculating the force due to pressure and the force due to viscosity, it is necessary to determine the amount of water represented by the heightfield that lies within the core radius of a given SPH particle. This section presents the algorithm used to solve this problem. It is important to note that in consideration of speed and ease of implementation, this algorithm only attempts to compute an approximation to this volume.

The SPH particle can be modeled as a sphere whose radius is equal to the core radius of the particle. The top of each column is modeled as a triangle, as described in section 11.2.1. The first step of the algorithm is to determine which triangular columns the particle is near. This is done by first dividing the x and y components of the particle's position by Δx and Δy respectively. This will indicate which set of grid points the particle resides between.

The next step is to project the sphere onto the plane of the heightfield simulation. Let's assume that the 2D heightfield grid is on the x-y plane. The projection is accomplished by simply setting the z-component of the particle's position to zero. The first part of the problem has now been reduced to determining which triangles in the heightfield simulation. This can be accomplished with a simple circle-triangle collision test. For an example see [8]. Now that the list of interacting columns has been determined the next step is to determine what quantity of volume lies within each column and the sphere representing the particle.

Each column of the heightfield simulation can be visualized as a triangular prism. The prism can be divided into an octree by recursively placing a new vertex mid-way along each edge, see Figure 11.4. The process of estimating the volume inside of the column and the particle becomes simply summing the volumes of the largest prisms in the octree whose vertices all lie within the core radius of the particle's center.

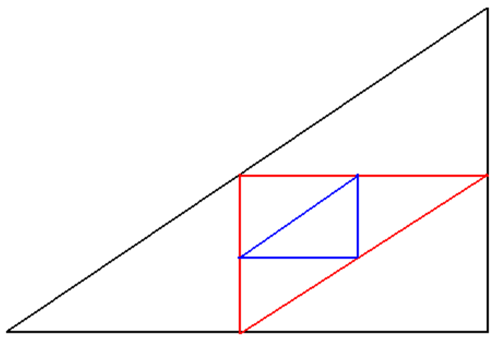


Figure 11.4: 2D visualization of the octree building process. In 2D the result is a quadtree. The extension to 3D is straightforward. The black triangle represents the base of the heightfield column. The red triangle represents the first level of the quadtree. The blue triangle represents a node at the next level.

11.2.6 Surface Extraction

The extraction of the surface will follow the color field approach discussed in section 8.1.3. While the color field is valid for the SPH particles it would ignore the surface given by the heightfield. Therefore, during the surface extraction process additional static particles will be introduced along the surface of each column as shown by the green particles in figure 11.5. This approach should be sufficient to capture the effect of overturning water.

One of the other desired effects of the proposed technique is to provide a physical justification for the generation of spray. The technique would able to handle this quite easily. Any time that the main volume of water is no longer within an SPH particle's core radius then the volume of water will no longer have an effect on the particle's motion and thus the particle can be thought of as spray. Once a spray particle has been identified then it will be subject to simple kinematics until it moves back within its core radius of the main water volume.

11.2.7 Interaction with Rigid Bodies

Including interaction with rigid bodies is also easily accomdated by this technique. When a rigid body makes contact with the surface of the water then its geometry will be projected along the object's velocity direction onto the water surface. Water from the height field simulation will be converted into SPH particles that will be uniformly distributed throughout the projected area. When the object contacts the particles the collision response can be handled in the same

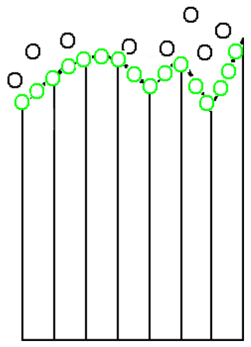


Figure 11.5: A 2D cross section of the proposed surface extraction technique. Here the black particles are the typical SPH particles and the green particles have been introduced along surface of the heightfield columns in order to account for the surface of the heightfield.

manner as any other collision between an object and a sphere.

If the density of the object is greater than that of the water then it will continue to sink. In this scenario then the above procedure can be repeated each time that the object contacts the surface of the height field. Once the object reaches the bottom of the simulation domain then the pressure in the columns containing the object will likely have reached zero. However, since the effect of the object on the surface will be negligible then it can be ignored and water is allowed to flow back into the columns either from neighboring columns or from SPH particles entering the column.

If the density of the object is less than that of the water then it should eventually come to rest on the surface. The particle-based collision response technique discussed above can be used to slow the object. Once the object's vertical velocity has fallen below a pre-determined threshold value, υ, then it can be treated as a simple particle that moves with the flow.

11.2.8 Converting Particles To and From the Heightfield Simulation

From Heightfield To Particles

As discussed in section 11.2.7, when a dynamic object comes near the surface of the heightfield then a thin layer along the surface of the water column needs to be converted from the heightfield simulation into particles. Let i and j be predetermined constants that describe the number of particles to introduce per unit width of the 2D grid and the number of particles to introduce per unit length of the 2D grid, respectively. When converting water from the heightfield into

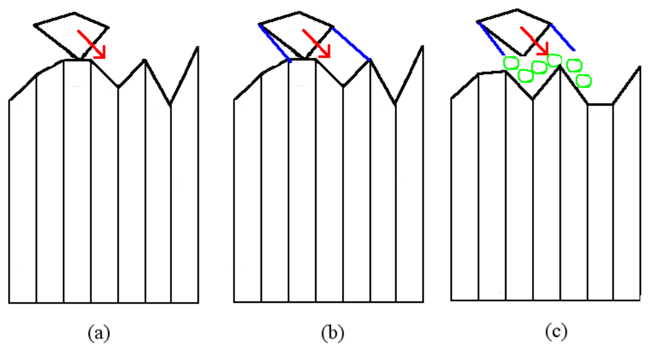


Figure 11.6: A 2D cross section of the proposed rigid body interaction technique. Part (a) represents the object coming into contact with the water surface. The red arrow is the object's velocity. Part (b) represents the projection of the object onto the water's surface. Part (c) shows the reduction of the volume in the columns being converted into particles inside of the object's projected area.

particles then the number of particles to be created, n, is equal to

$$n = (i * \Delta x)(j * \Delta y) \tag{11.2.17}$$

where Δx is the grid spacing in the x-direction and Δy is the grid spacing the in y-direction. The particles are evenly distributed throughout the area on the surface of the column. The velocity of each particle is taken as the average of the velocities at each of the three points comprising the column. Lastly, the mass of each particle needs to be determined.

Let the user-defined constant, Δv, represent the amount of volume to be converted into particles. This value of Δv is used any time that a thin layer of water along the top of the heightfield needs to be converted into particles. If the volume in a given column is less than v then let $v = v_c$ where v_c is the volume of the column. Using n and Δv then the mass of each of the particles is simply

$$m_p = \frac{\Delta v}{n} \tag{11.2.18}$$

When water is converted into particles, then the volume in the column needs to be reduced. The change in pressure at each grid point, δp_c, can be obtained by taking the derivative of equation (11.2.4)

$$-\Delta v = s * \frac{1}{2}\left(\frac{1}{3}\right)(\delta p_1 + \delta p_2 + \delta p_3)\,\Delta x \Delta y \tag{11.2.19}$$

If we enforce the constraint that the change in pressure be equal at each grid point, that is $\delta p_1 = \delta p_2 = \delta p_3 = \delta p_i$, then (11.2.19) becomes

$$\begin{aligned} -\Delta v &= s * \frac{1}{2}\left(\frac{1}{3}\right)(\delta p_i + \delta p_i + \delta p_i)\,\Delta x \Delta y \\ -\Delta v &= s * \frac{1}{6}(\delta p_i + \delta p_i + \delta p_i)\,\Delta x \Delta y \\ -\Delta v &= s * \frac{1}{2}\delta p_i \Delta x \Delta y \\ \frac{-2\Delta v}{s \Delta x \Delta y} &= \delta p_i \end{aligned} \tag{11.2.20}$$

here Δv is negative because it represents volume being removed from the column.

From Particles To Heightfield

Collision between a particle and the heightfield can easily be determined by checking the height of the water surface at the point where the particle resides. If the particle's height is less than or equal to the height of the surface, then it is colliding with the water. If the particle is colliding with the surface and the distance from the particle to any static or dynamic object is greater than a predetermined threshold, d, then the particle will be converted back into the heightfield simulation. Otherwise, the particle will remain in its current position and be updated with the simulation.

Once the need to convert a particle into the heightfield has been determined, then the volume represented by the particle is added into the column on which it resides. If a particle resides on the boundary between one or more columns, then its volume is spread evenly across all columns. The change in pressure for a column that is absorbing a particle is the same as equation (11.2.19) except that a positive value of Δv is used. If a particle resides over n columns, then the volume added to each column is simply $\Delta v = v_p/n$ where v_p is the volume of water represented by the particle.

BIBLIOGRAPHY

[1] Allen, M., and Wilkinson, B. *Parallel Programming.* Prentice Hall, Inc, Upper Saddle River, New Jersey, 1999.

[2] Allstar network. aeronautics - fluid dynamics - level 3 - flow equations. `http://www.allstar.fiu.edu/aero/Flow2.htm`, January 29th, 2007.

[3] B.D. Nicholas, C. W. H. Calculating three-dimensional free surface flows in a vicinity of submerged and exposed structures. *Journal of Computational Physics*, 12 (1973).

[4] Cfd online. gradient computation. `http://www.cfd-online.com/Wiki/Gradient_computation`, 2 February 2007.

[5] Chen, J. X., and da Vitoria Lobo, N. Toward interactive-rate simulation of fluids with moving obstacles using navier-stokes equations. *Graph. Models Image Process. 57*, 2 (1995), 107–116.

[6] David Mould, Y.-H. Y. Modeling water for computer graphics. *Computers and Graphics 21*, 6 (November 1997), 801–814.

[7] D.J. Auld, K. S. Flow description, streamline, pathline, streakline and timeline. `http://www.ae.su.oz.au/aero/fprops/cvanalysis/node8.html`, November 16, 2006.

[8] Ericson, C. *Real Time Collision Detection.* Morgan Kaufman, 2005.

[9] Fedkiw, R. P., Aslam, T., Merriman, B., and Osher, S. A non-oscillatory eulerian approach to interfaces in multimaterial flows (the ghost fluid method). *J. Comput. Phys. 152*, 2 (1999), 457–492.

[10] Flow science inc. weir flow. `http://www.flow3d.com/Appl/weir.htm`, December 02, 2006.

[11] Foster, N., and Fedkiw, R. Practical animation of liquids. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), ACM Press, pp. 23–30.

[12] Foster, N., and Metaxas, D. Realistic animation of liquids. *Graph. Models Image Process. 58*, 5 (1996), 471–483.

[13] Giancoli, D. C. *Physics for Scientists and Engineers*, 3 ed. Prentice Hall. Upper Saddle River, New Jersey, 2000.

[14] Holmberg, N., and Wünsche, B. C. Efficient modeling and rendering of turbulent water over natural terrain. In *GRAPHITE '04: Proceedings of the 2nd international conference on Computer graphics and interactive techniques in Australasia and South East Asia* (New York, NY, USA, 2004), ACM Press, pp. 15–22.

[15] *IRIX Device Driver Reference Pages.* Silicon Graphics, Inc, November 14, 1994. `http://techpubs.sgi.com/library/tpl/cgi-bin/getdoc.cgi/0530/bks/SGI_Developer/books/DevDriver_PG/sgi_html/apa.html#id35547`.

[16] Jiun-Der Yu, Shinri Sakai, J. A. S. A coupled level set projection method applied to ink jet simulation. *Interfaces and Free Boundaries 5* (2003), 459–482.

[17] Kelager, M. Lagrangian fluid dynamics using smoothed particle hydrodynamics. Master's thesis, University of Copenhagen, 2006.

[18] Kipfer, P., and Westermann, R. Realistic and interactive simulation of rivers. In *GI '06: Proceedings of the 2006 conference on Graphics interface* (Toronto, Ont., Canada, Canada, 2006), Canadian Information Processing Society, pp. 41–48.

[19] Liu, G., and Liu, M. *Smoothed Particle Hydrodynamics.* World Scientific, 2003.

[20] LMNO Engineering, R., and Software, L. Rectangular weir calculator. `http://www.lmnoeng.com/Weirs/RectangularWeir.htm`, December 02, 2006.

[21] Martin, T. J., Pearce, F. R., and Thomas, P. A. An owner's guide to smoothed particle hydrodynamics. *ArXiv Astrophysics e-prints* (Oct. 1993).

[22] Matthews, P. *Vector Calculus.* Springer, 1998.

[23] Müller, M., Charypar, D., and Gross, M. Particle-based fluid simulation for interactive applications. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2003), Eurographics Association, pp. 154–159.

[24] Sethian, J. A. Level set methods: An act of violence. *American Scientist* (May-June 1997).

[25] Song, O.-Y., Shin, H., and Ko, H.-S. Stable but nondissipative water. *ACM Trans. Graph. 24*, 1 (2005), 81–97.

[26] Stam, J. Stable fluids. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1999), ACM Press/Addison-Wesley Publishing Co., pp. 121–128.

[27] Temam, R. Une mthode d'approximation de la solution des quations de navier-stokes. *Bull. Soc. Math. de France*, 98 (1968), 115–152.

[28] Wikipedia. Fick's law of diffusion. `http://en.wikipedia.org/wiki/Fick%27s_law`, 12 September 2006.

[29] Wikipedia. Navier-stokes equations. `http://en.wikipedia.org/wiki/Navier-Stokes_equations`, November 16, 2006.

[30] Wikipedia. Reynolds number. http://en.wikipedia.org/wiki/Reynolds_number, November 16, 2006.

[31] William H. Press, Saul A. Teukolsky, W. T. V., and Flannery, B. P., Eds. *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, 1992.

[32] Zwicker, M., Pfister, H., van Baar, J., and Gross, M. Surface splatting. In *SIGGRAPH 2001, Computer Graphics Proceedings* (2001), E. Fiume, Ed., ACM Press / ACM SIGGRAPH, pp. 371–378.