STYLISTIC CONTROL OF OCEAN WATER SIMULATIONS

A Thesis

by

CHRISTOPHER WAYNE ROOT

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

December 2008

Major Subject: Visualization Sciences

STYLISTIC CONTROL OF OCEAN WATER SIMULATIONS

A Thesis

by

CHRISTOPHER WAYNE ROOT

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

Chair of Committee,	Donald House
Committee Members,	John Keyser
	Richard Davison
Head of Department,	Tim McLaughlin

December 2008

Major Subject: Visualization Sciences

ABSTRACT

Stylistic Control of Ocean Water Simulations. (December 2008)

Christopher Wayne Root, B.S., University of Minnesota

Chair of Advisory Committee: Dr. Donald House

This thesis presents a new method for controlling the look of an ocean water simulation for the purpose of creating cartoon-styled fluid animations. Two popular techniques to simulate fluid, a statistical height field method via the Fast Fourier Transform and the Stable Fluid method for dynamic effects, are connected smoothly via a blend domain, thus allowing a height field to drive a physical simulation. In addition, the height field can be stylized by utilizing a keyframing technique on wave amplitudes defined in the Fourier domain, allowing for creative control of the fluid's surface. Such stylized height fields therefore can be simulated to exhibit natural fluid motion as well as to produce dynamic effects such as breaking waves that were previously unattainable in common fluid pipelines.

NOMENCLATURE

CLSVOF	Combined Level Set and Volume-of-Fluid
FFT	Fast Fourier Transform
MAC	Marker-and-Cell
NURBS	Non-Uniform Rational B-Spline
PLS	Particle Level Set
VOF	Volume-of-Fluid

TABLE OF CONTENTS

Page

LIST OF FIGURES

1 INTRODUCTION

Fluid effects have been widely used in animated films for decades. Classic Disney films such as *Alice in Wonderland*, *Pinocchio*, & *Fantasia* used fluid effects because "[they] had become an integral part of the film, contributing drama and excitement and mood, as well as the vital element of making everything so believable" [17]. Because of this importance, specialized animators were employed solely to create natural effects like fog, dust, and fluid. Realistic fluid, however, turned out to be terribly difficult for them to animate. As Frank Thomas, one of Walt Disney's original team of animators known as the "Nine Old Men," put it, "The combination of transparency, elasticity, weight, mobility, and consistency … made it impossible to handle [water] realistically" [17]. Therefore stylizing the fluid, while maintaining believability, became much more crucial than achieving natural realism.

Today, much of the effects animation traditionally drawn by hand is controlled by computer simulations thanks in large part to a spate of recent computer graphics research. However, much of this research strives for realism. Little exploration has been done to stylize the look of a 3D simulation, specifically that of water. Even modern animated films with fluid elements like Pixar's *Finding Nemo*, which are arguably influenced by their classic 2D counterparts, use very realistic water.

This thesis proposes to use common simulation techniques to render stylized

---

The journal model is *IEEE Transactions on Visualization and Computer Graphics.*

fluid inspired by classic Disney films. Stylization is primarily obtained by using a keyframing technique on wave amplitudes defined in the frequency domain which are subsequently converted into a height field. This allows one to create stylized surfaces that exhibit believable fluid motion.

However, such a height field cannot be used alone if dynamic effects like breaking waves are intended. In fact, there has yet to be found an all-encompassing simulation technique that will practically handle all fluid dynamics from open water to breaking waves. This problem is addressed by incorporating a unique blend domain allowing a height field to drive a physical simulation to produce dynamic effects practically in a modern animation production environment.

2 PRIOR WORK

Computational fluid dynamics has been an actively studied area of research for centuries. In fact, many fluid techniques employed by modern-day computer graphics pipelines are derived directly from very early studies. Fournier and Reeves' [9] work for example, which is considered one of the first to develop a fluid model for computer graphics, is based on the Gerstner model [11] developed in 1802 which determines that the motion of an individual particle of water due to a wave follows a circular or elliptical orbit. The Fournier/Reeves' model generates the surface of the water using a displacement approach that exhibits this type of motion, and can also be affected by natural phenomena such as wind, gravity, and depth.

Statistical models based on empirical observation of the ocean tend to be used more often for fully-developed seas. Such models, initially developed by Matsin, et al. [22] and further examined by Tessendorf [30], use analytical spectral data to create a wave height representation of the ocean surface in the Fourier domain. Height fields are easily generated via Fast Fourier Transforms and have produced very realistic effects in films such as *Titanic*, *Waterworld*, and *The Perfect Storm*.

However, both the Gerstner and statistical models of fluid surfaces are unable to handle complex fluid flow such as breaking waves. For such dynamic effects, techniques based on the full Navier-Stokes equations for fluid flow are employed. Foster and Metaxas [8] utilized the marker and cell (MAC) method of Harlow and Welch [14] to solve these equations in an eulerian-discretized framework suitable for computer

graphics. Stam [28] later introduced an unconditionally stable semi-lagrangian method to solve the momentum conservation step of the equations allowing for much larger time intervals in the simulation. This technique can also be easily adapted to advect any kind of smooth quantity in a velocity field, again with the benefit of it never reaching instability.

To track the fluid's surface, Foster and Metaxas [8] primarily used marker particles. However, generating a smooth realistic fluid surface from particles alone is very difficult. Mihalef, et al. [25] coupled a level set implicit surface with a volume-of-fluid technique which stores the fluid volume in any given cell to track the surface. This technique is very successful at preserving the overall volume of the fluid, but suffers from numerical error that results in unwanted “blobbies”. Foster and Fedkiw [7] introduced a hybrid fluid volume model that combines a level set with marker particles to help correct numerical dissipation due to advecting the surface in the velocity field. Enright et al. [5] extended this hybrid approach by placing marker particles on both sides of the fluid surface thereby further correcting the dissipation. They also provided a means to define valid velocities in the air which the particles can effectively use. This *Particle Level Set Method* results in a smooth surface that can capture fine fluid detail. A byproduct of using these marker particles is that they can be used to generate other fluid effects such as bubbles (Greenwood & House [12], Cleary et al. [2]), spray (Guendelman et al. [13], Kim et al. [19]), and foam (Kim et al. [18]).

Controlling the behavior of fluid flow is an increasing area of study, and yet very challenging. Some recent work has been quite successful at controlling the behavior of

smoke, such as Treuille et al. [31], Fattal and Lischinski [6], and McNamara et al. [23]. The latter makes use of an efficient gradient-calculating technique known as the *adjoint method* which can be used to control level-set fluids in addition to smoke. However, thanks to a level set fluid's many discontinuities, control is much more difficult due to an increased inaccuracy of the gradient. Mihalef, et al. [25] generated some nice controlled breaking wave simulations using a slice method. Each slice is represented by a 2D wave profile simulation that can be used to direct the final look of a 3D wave. However, such an approach is limited to a wave break only and is not easily extended to other areas of the fluid. It is also limited by the directability of the 2D profile simulations, therefore a cartoon-style wave may not be possible if such a look cannot be achieved in 2D.

3 METHODOLOGY

In order to create a stylized fluid simulation with natural dynamic effects, our method relies on an organization of the simulation domain into three spatial domains as shown in Figure 1. They are the:

1. Statistical Domain: maintains a user-controlled height field
2. Physical Domain: solves the Navier-Stokes equations for a physically accurate fluid simulation
3. Blend Domain: performs blending techniques for a smooth transition from the statistical to physical domains

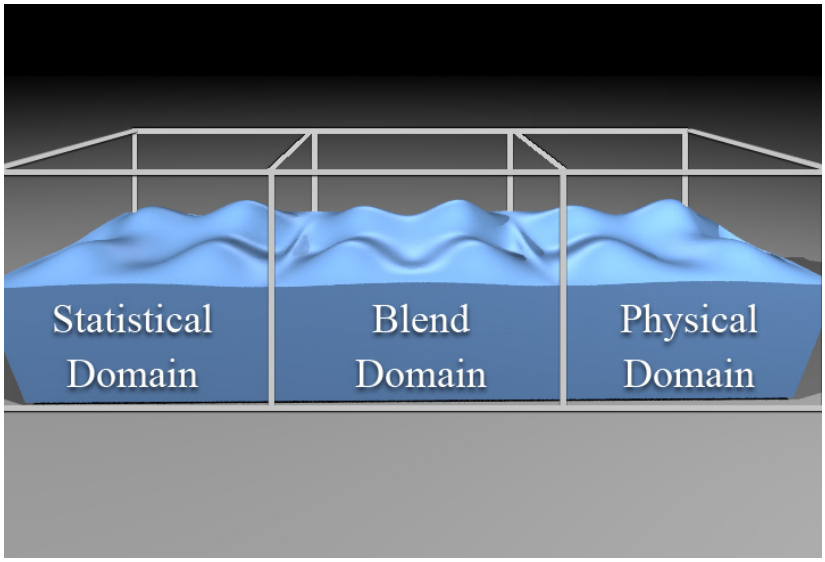

Fig. 1. Simulation Domains.

This section will continue as follows: first, methods to generate and control waves in the statistical domain are discussed. Second, we describe means by which the Navier-Stokes equations are solved in the physical domain, followed by a discussion of how blending between the two is performed in the blend domain. Finally we present methods by which the fluid can be rendered in a stylized manner.

3.1 Wave Generation

Given the challenge of creating directable cartoon-like ocean simulations, efforts were focused on creating waves that could be controlled by the artist, yet exhibited natural fluid motion. Two approaches of generating waves were examined: the Gerstner model [9] and the statistical height field model [30]. After careful analysis, the statistical model deemed more appropriate for three key reasons:

1. The fluid will always exhibit natural motion due to a wave dispersion relation that is independent of its surface height.
2. Only one degree of control is needed: the wave amplitudes, thus making it very straight forward for a user to generate and control a height field. By contrast, the Gerstner model requires a wave vector, amplitude, frequency, and phase for each wave. All of these but the amplitudes are predetermined in the statistical model thanks to mathematical properties present after a height field is decomposed into a sum of sine and cosine waves via the FFT.
3. Due to a property of the Fourier transform, the resulting height field will be

totally periodic allowing for a very direct way to initialize both the fluid surface and velocities in the neighboring domains.

We based our statistical model on that described by Tessendorf in [30]. His method is derived from empirical observations of ocean water which found that waves exhibit nice spectral properties that can be defined in the Fourier, or frequency, domain. A Fourier-based representation of a wind-driven height field in open water is therefore defined as:

$$h(\mathbf{x},t) = \sum_{\mathbf{k}} \tilde{h}(\mathbf{k},t)\exp(i\mathbf{k}\cdot\mathbf{x}) \tag{3.1}$$

where $\mathbf{x}$ is a horizontal positon, t is time, $\mathbf{k}$ is a wave vector defining a phase and direction of a wave, and $\tilde{h}(\mathbf{k},t)$ are the complex, time-dependent amplitudes. This sum can be rapidly computed at discrete, evenly-spaced points using an FFT.

3.1.1 Wave Amplitudes

For a realistic height field, wave amplitudes are generated according to oceanographic statistical observations which show them to be nearly independent, gaussian fluctuations with a spatial spectrum defined by:

$$P_h(\mathbf{k}) = \left\langle \left| \tilde{h}^*(\mathbf{k},t) \right|^2 \right\rangle \tag{3.2}$$

for some spectral model $\tilde{h}^{*}(\mathbf{k},t)$. For example, a useful model commonly used for realistic wave generation in open, wind-driven water is the *Phillips Spectrum*:

$$P_h(\mathbf{k}) = A\frac{\exp(-1/(kL)^2)}{k^4}\left|\hat{\mathbf{k}}\cdot\hat{w}\right|^2 \tag{3.3}$$

where k is the wavenumber, or magnitude of the wave vector, $L = V^2/g$ is the largest possible wave from a continuous wind with speed V, g is the gravitational constant, normally $9.8\ m/sec^2$, and $\hat{w}$ is the wind direction.

Of course other spectral functions can be used to achieve desired effects. For instance, to get only waves that travel in the direction of the wind, the *Phillips Spectrum* can be modified like so:

$$P_h(\mathbf{k}) = A\frac{\exp(-1/(kL)^2)}{k^4}\max(\left|\hat{\mathbf{k}}\cdot\hat{w}\right|,0)^2 \tag{3.4}$$

A comparison of these two spectral functions can be seen in Figure 2.

After a spectral function has been defined, the fourier wave amplitudes at time $t = 0$ can be generated as follows:

$$\tilde{h}_0(\mathbf{k}) = \frac{1}{\sqrt{2}}(\xi_r + i\xi_i)\sqrt{P_h(\mathbf{k})} \tag{3.5}$$

where ξ_r and ξ_i are guassian random numbers with a mean of 0 and a standard deviation of 1.

3.1.2 Dispersion

Water waves have a well known relationship between their frequencies and the magnitude of their wave vectors. This relationship in deep water where the ocean floor can be ignored is:

$$\omega^2(k) = gk \tag{3.6}$$

where g is the gravity constant, and k is the wave number. This is commonly known as

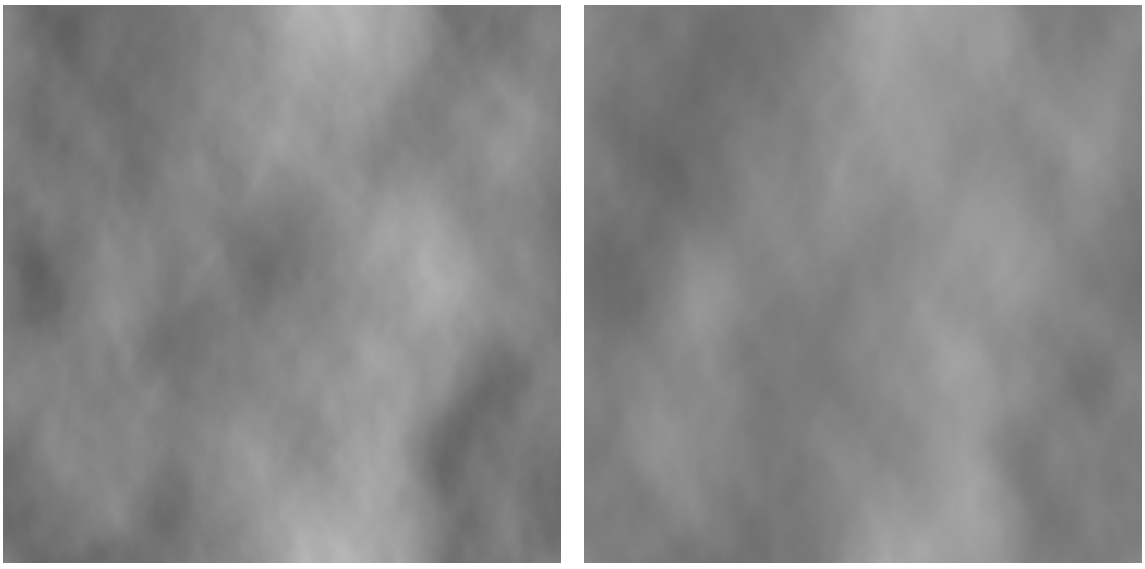

Fig. 2. Height Field Image Comparison. Height field images of eqns. (3.3) and (3.4) respectively using a 512x512 domain. Notice in the second image that much detail is lost and that the remaining waves face the direction of the right-blowing wind.

the dispersion relation. There are other similar dispersion relation equations that have been developed for shallow water and for detailed fluids where surface tension is a factor, but for the purposes of this thesis it can be safely assumed that the above dispersion function will suffice.

With a dispersion relationship defined, the Fourier amplitudes at time *t* are:

$$\begin{aligned}\tilde{h}(\mathbf{k},t) &= \tilde{h}_0(\mathbf{k})\exp\{i\omega(k)t\} \\ &\quad + \tilde{h}_0^*(-\mathbf{k})\exp\{-i\omega(k)t\}\end{aligned} \tag{3.7}$$

This form preserves the complex conjugation property needed for the FFT to be applied and allows waves to propagate in multiple directions.

3.1.3 User-Determined Heights

After establishing initial conditions for the physical size of the statistical domain, the number of discrete samples to use, and a wind speed and direction, a height field can be built which the user can use as a template to model their own height field. Modeling can be performed on any standard 3D surface used to represent the height field, but for the purpose of this thesis NURBS curves and surfaces [26] worked well due to their smooth surface representation (good for cartoon-like fluid) and reasonable local control. An initial NURBS surface with a specified number of control points is created by using a least-squares method available via an open-source nurbs++ library [20] to approximate the height field. The user can then model the NURBS surface to represent a height field of their choice.

Once modeling is complete, wave amplitudes can be extracted in a partially inverse way to the procedure described in sub-section 3.1.1. The NURBS surface will be sampled at specific equidistant points as determined by initial sampling conditions. These sampled heights are then converted into a Fourier representation using the FFT. By enforcing a restriction that this transformation is always applied at time $t = 0$, the resulting Fourier coefficients will be $\tilde{h}(\mathbf{k}, 0)$ or in other words:

$$\tilde{h}(\mathbf{k}, 0) = \tilde{h}_0(\mathbf{k}) + \tilde{h}_0^*(-\mathbf{k}) \tag{3.8}$$

Half of the initial height amplitudes $\tilde{h}_0(\mathbf{k})$ will be defined using (3.5). However, thanks to the complex conjugation property, the other half are simply:

$$\tilde{h}_0^*(-\mathbf{k}) = \tilde{h}(\mathbf{k}, 0) - \tilde{h}_0(\mathbf{k}) \tag{3.9}$$

3.1.4 Keyframe Control

A height field can be keyframed at any point in time using this approach. After an initial NURBS surface has been created, the user can playback its resulting motion, and if at any point they wish to change its look, they can do so by modifying the surface and adding a new wave amplitude keyframe. These keyframed amplitudes will be generated in the same way as described in 3.1.3 and the resulting height field will be a result of interpolating them in the Fourier domain. An example of the keyframe control that can be applied to fluid surfaces in our method is shown in Figures 3 and 4.

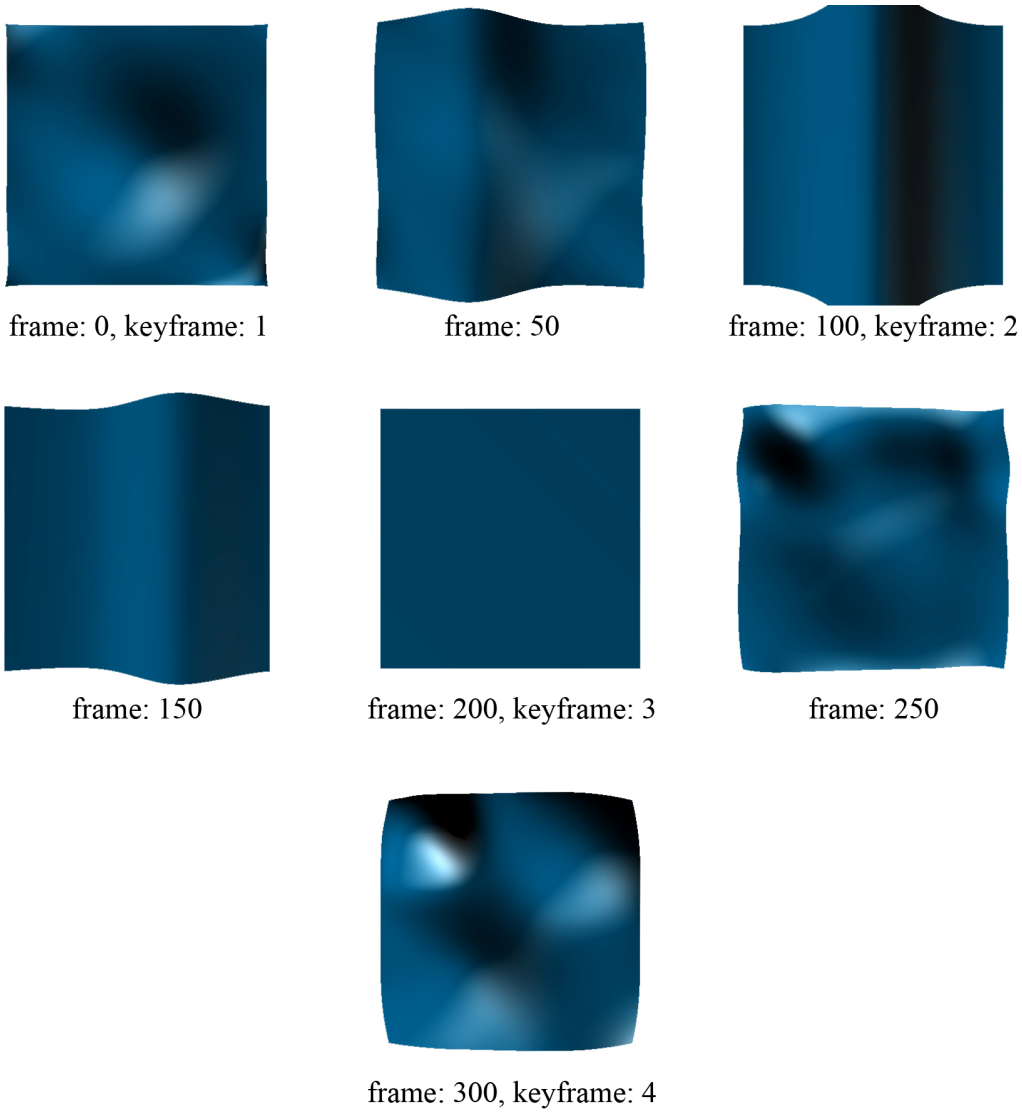

Fig. 3. Ocean Surface Keyframes I. The surfaces marked as a keyframe are user-defined. The others are interpolated surfaces at a frame halfway between successive keyframes.

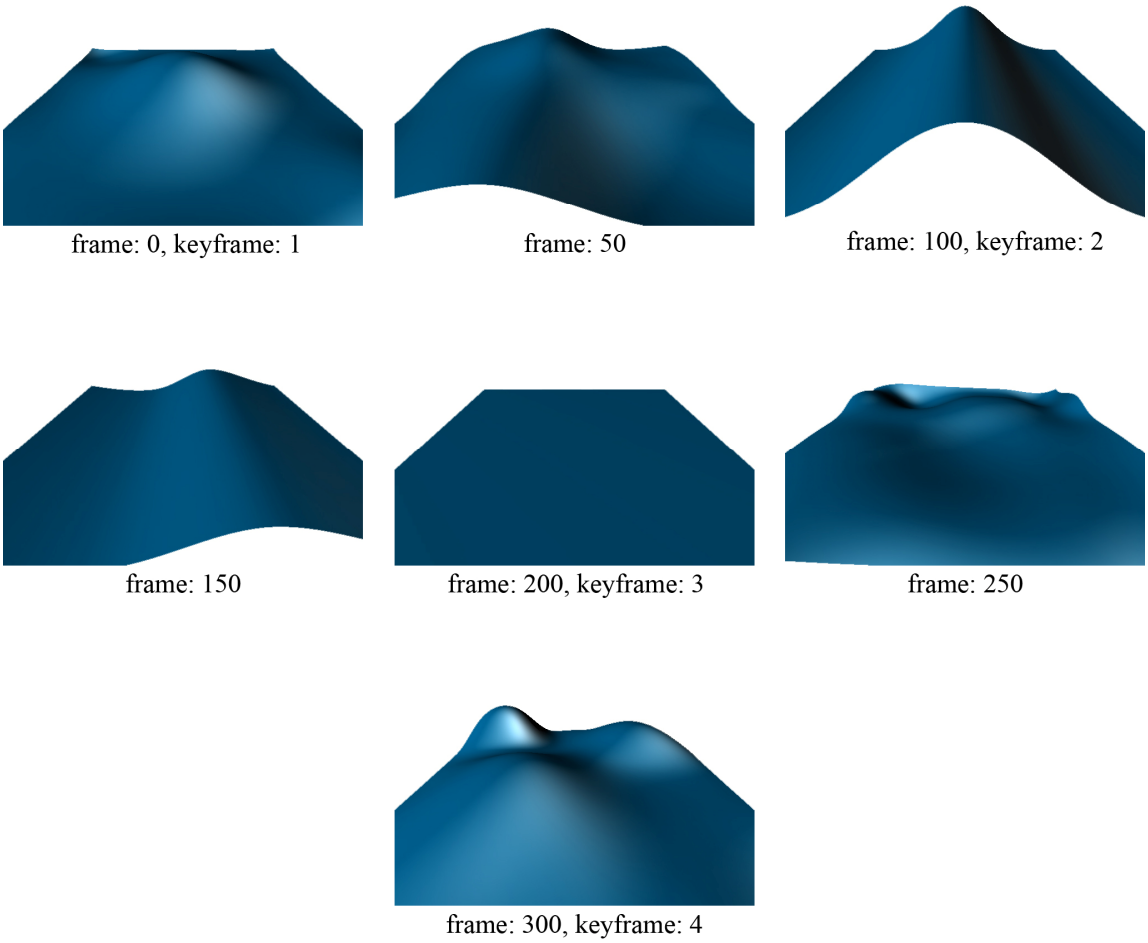

Fig. 4. Ocean Surface Keyframes II. Same as Fig. 3, but from a different camera angle.

3.2 Dynamic Fluid

To solve the dynamically changing fluid in the physical domain of the simulation, the commonly used “stable fluid” method introduced by Stam [28] was employed. This method describes an unconditionally stable solver for the incompressible Navier-Stokes equations:

$$\mathbf{u}_t = -(\mathbf{u} \cdot \nabla)\mathbf{u} - \frac{1}{\rho}\nabla p + v\nabla^2\mathbf{u} + \mathbf{f} \tag{3.10}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{3.11}$$

Equation (3.10), the momentum equation, determines the acceleration of a fluid with velocities $\mathbf{u}$, pressures p, density ρ, viscosity v, and external forces $\mathbf{f}$. Equation (3.11) is the incompressibility condition which constrains the fluid's volume to stay constant.

3.2.1 Stable Fluid Method

Equation (3.10) can be solved numerically by splitting the equation up into its component parts and solving each part separately in turn [1]. Splitting the momentum equation (3.10) for an inviscid fluid results in 3 intermediate equations:

1. $\frac{\partial \vec{u}}{\partial t} + \nabla \vec{u} \cdot \vec{u} = 0$ (*advection*) (3.12)

2. $\frac{\partial \vec{u}}{\partial t} - \vec{g} = 0$ (*external forces*) (3.13)

3. $\frac{\partial \vec{u}}{\partial t} + \frac{1}{\rho}\nabla p = 0$ such that $\nabla \cdot \vec{u} = 0$ (*pressure/incompressibility*) (3.14)

Due to the fact that water has very little viscosity and that our numerical solution for the above equations inherently introduces error that can be reinterpreted as such, the viscosity term can be dropped.

To efficiently solve these split equations for purposes of computer graphics, the entire simulation domain is discretized by a MAC grid as shown in Figure 5 [1]. The velocities $\vec{u}$ are defined at the wall centers of the cell and any other constant or advectable quantities such as pressure p are defined at the center of the cell. This grid structure is used to provide an accurate central difference for calculating spatial derivatives in the split equations.

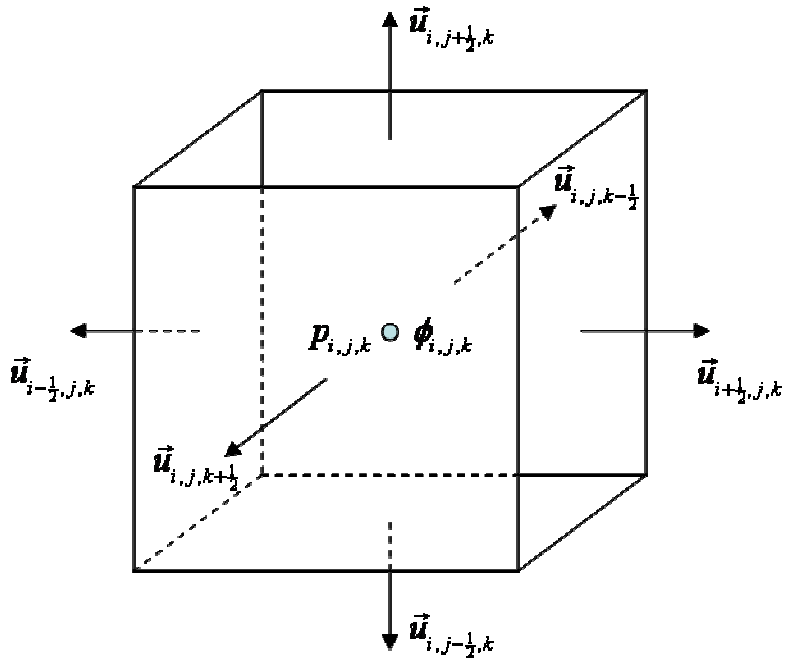

Fig. 5. 3D MAC Grid Cell. Used to discretize the simulation domain.

The equations are then solved as follows. First, to preserve momentum, we advect the divergence-free velocity field $\mathbf{W}^0$ that is a result of solving the Navier-Stokes equations in the previous time step. Advection is carried out using Stam's semi-Lagrangian technique which finds a new velocity at each grid sample point by integrating backwards in time as if a particle was advected in the velocity field and ended up at the sample point in question. Our method makes use of an accurate 4^{th} order Runga-Kutta integration scheme to trace the velocity sample backwards in time and results in an intermediate velocity field $\mathbf{W}^1$. The advection equation (3.12) could be solved alternatively by using an accurate central difference for the spatial derivative and a high-order integration scheme for the time derivative, but it will eventually become unstable regardless of the time step used. A small time step would only delay the inevitable instability. However, Stam's semi-Lagrangian technique is unconditionally stable for any time step, which makes it a very desirable method for our purposes.

We then apply any outside forces, like gravity and wind, to the intermediate velocity field $\mathbf{W}^1$ by using a 1^{st} order Forward Euler integration scheme for the time derivative in eq. (3.13). In other words:

$$\mathbf{W}^2 = \mathbf{W}^1 + \Delta t \mathbf{f} \tag{3.15}$$

Finally we solve for pressures and incompressibility in eq. (3.14) again using Forward Euler:

$$\mathbf{W}^3 = \mathbf{W}^2 - \Delta t \frac{1}{\rho} \nabla p \tag{3.16}$$

By enforcing the incompressibility contstraint, eq. (3.16) can be re-written as:

$$\nabla \cdot \mathbf{W}^3 = \nabla \cdot \mathbf{W}^2 - \Delta t \frac{1}{\rho} \nabla^2 p = 0 \tag{3.17}$$

This produces a large system of equations for pressure p that can be solved using a bi-conjugate gradient method.

In summary, the Navier-Stokes equations for incompressible fluid flow is solved at each time step by doing the following:

$$\begin{aligned} \mathbf{W}^0 &= \vec{u}^n \\ \mathbf{W}^1 &= advect(\mathbf{W}^0, -\Delta t) \\ \mathbf{W}^2 &= \mathbf{W}^1 + \Delta t \mathbf{f} \\ \nabla^2 p &= \Delta t \frac{1}{\rho} \nabla \cdot \mathbf{W}^2 \\ \vec{u}^{n+1} &= \mathbf{W}^2 - \Delta t \frac{1}{\rho} \nabla p \end{aligned} \tag{3.18}$$

3.2.2 Tracking the Fluid Surface

Determining the location of the surface of the fluid is not only important to account for fluid/air boundaries in the stable fluid method, but it's also vitally important for rendering. For the purposes of this thesis, two approaches to track the fluid interface

were studied, the Coupled Level Set and Volume-of-Fluid (CLSVOF) method introduced by Sussman and Puckett [29], and the Particle Level Set (PLS) method by Enright et al. [5]. Both techniques make use of an implicit surface (known as a level set) which is defined by a signed distance function ϕ where negative ϕ values are found inside the surface, positive values are outside, and the surface boundary is at $\phi = 0$. The level set is advantageous because of inherent merging properties of an implicit surface and the ease by which it can be advected in a surrounding velocity field.

The level set advection equation is:

$$\frac{\partial \phi}{\partial t} + \nabla \phi \cdot \vec{u} = 0 \tag{3.19}$$

Notice that this equation takes on the same form as the velocity advection equation (3.12) in the stable fluid method. Therefore it can also be solved using a semi-Lagrangian technique utilizing unconditional stability. However, level sets suffer from numerical dissipation due to the interpolated averaging needed as a result of discretizing on an eulerian MAC grid. Such dissipation results in loss of interesting fluid flow and volume loss.

One way to reduce this error is by maintaining the signed distance propery for the level set:

$$\left|\nabla \phi\right| = 1 \tag{3.20}$$

Maintaining such a property significantly increases the accuracy whenever spatial differentiation is needed. Of course, as a level set is advected, the signed distance property won't hold, so it is maintained by performing a Fast Marching Method as described by Sethian [27] after advection. However, maintaining the signed distance property won't correct dissipation. It will only help improve advection accuracy. Therefore level sets have been coupled with other techniques to improve interface capturing as the level set moves. We chose to make use of the PLS method [5] over the CLSVOF method [29] because special consideration needs to be taken when advecting the VOF function since it is not a smooth-varying function as shown in Figure 6.

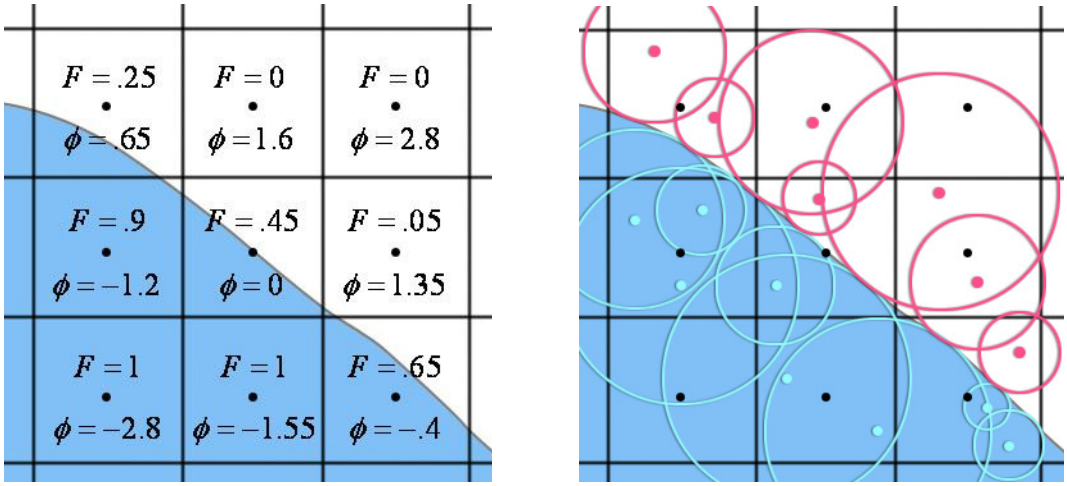

Fig. 6. Comparison of a 2D CLSVOF and PLS Grid. Notice that the VOF function *F* is not a smooth-varying function in the first image.

3.2.3 Particle Level Set Method

The PLS method implemented is based on the techniques described in [4]. It is a "thickened" hybrid approach to tracking the interface of an implicit level set. It makes use of marker particles randomly placed on both sides of the interface to help correct numerical dissipation, thus preserving detailed fluid flow and volume. Both positive and negative particles are placed within some small distance about the surface and given a radius determined by the following:

$$r_p = \begin{cases} r_{max} & if \;\; s_p\phi(\vec{x}_p) > r_{max} \\ s_p\phi(\vec{x}_p) & if \;\; r_{min} \le s_p\phi(\vec{x}_p) \le r_{max} \\ r_{min} & if \;\; s_p\phi(\vec{x}_p) < r_{min} \end{cases} \tag{3.21}$$

where s_p is the sign of the particle, $\phi(\vec{x}_p)$ is the value of the level set function at the particle's position, and r_{min} and r_{max} are the minimum and maximum particle radii. Typical values used for these radii are 0.1 and 0.5 times the size of a grid cell.

After the level set has been advected, particles are transported in the velocity field by using a 4th order Runga-Kutta integration scheme. Their new positions are then used to correct the level set. If a positive particle is found to have moved across the surface a distance more than its radius, it is labeled as having escaped, indicating that the level set may need correcting, likewise for negative particles. Correction is done by applying a simple spherical level set function to each particle as defined by:

$$\phi_p(\vec{x}) = s_p(r_p - \left|\vec{x} - \vec{x}_p\right|) \tag{3.22}$$

When grouped together closely, these individual particle level set functions will overlap, defining a better location for the fluid surface. Escaped negative particles are used to rebuild $\phi \le 0$ region and likewise escaped positive particles rebuild the $\phi > 0$ region by using local minima and maxima respectively:

$$\phi^- = \min_{\forall p \in E^-}(\phi_p, \phi^-) \tag{3.23}$$

$$\phi^+ = \max_{\forall p \in E^+}(\phi_p, \phi^+) \tag{3.24}$$

Finally, the level set is reconstructed by merging the rebuilt positive and negative regions according to the following condition:

$$\phi = \begin{Bmatrix} \phi^+ & if & \left|\phi^+\right| \le \left|\phi^-\right| \\ \phi^- & if & \left|\phi^+\right| > \left|\phi^-\right| \end{Bmatrix} \tag{3.25}$$

After merging these regions, the level set is re-initialized to a signed distance function using the Fast Marching Method [27] following which error correction is again performed since the level set could have moved slightly. Finally, the particles' radii are readjusted using eq. (3.21).

During the simulation, particle densities may be such that some areas about the interface may not have enough particles for accurate reconstruction. Therefore particles

will need to be periodically re-seeded to maintain proper densities. This can be determined by a user-defined value, but experimentation has shown that reseeding the particles every 20 frames was sufficient [4].

3.3 Blending of the Statistical and Physical Domains

Now that we have methods to provide user control of a height field and an accurate physical fluid solver, techniques to combine the two domains were explored with the idea that the height field could drive a physical simulation for added dynamic effects. This was accomplished by means of a blend domain inside which velocities extracted from the height field were merged with divergence-free velocities resulting from our stable fluid solver.

This sub-section will proceed as follows: first a way to convert the height field into a level set and initialization of the level set in the blend and physical domains are described. Second, we detail a method by which velocities can be extracted from a height field in the statistical domain. Finally, minor modifications to the stable fluid method are discussed and the algorithm for simulating the entire fluid is summarized.

3.3.1 Height Field-to-Level Set Conversion and Initialization

One very useful property of the height field generated via the FFT is that it is totally periodic. This is very convenient for it provides a way to initialize a level set in both the blend and physical domains regardless of their size since the level set can be duplicated until the domains are filled.

In the statistical domain, height values will be defined at discrete, evenly spaced sample points after performing an inverse FFT on a set of keyframe-interpolated wave amplitudes. These height values are used to generate a surface - a NURBS surface in our case. A NURBS surface can be built to accurately fit these sample height points via a global interpolation method provided by the nurbs++ library [20].

Now that a surface is defined in the statistical domain, we convert it to a level set so that we can easily initialize a surface in the blend and physical domains. We accomplish this by first initializing the ϕ level set values in MAC grid cells closest to the NURBS surface. Once these initial values are found, they can be used to fill in the rest of the level set field using a Fast Marching Method [27].

Close cells are determined by first projecting the grid sample points onto the NURBS surface. If the distance from the sample point to the resulting projection point is less than the size of a grid cell, it is considered a close cell. One might assume that we could just use this distance to initialize ϕ here too. However, this would not be accurate and could result in unwanted artifacts. Instead, we find the true closest distance to the NURBS surface by doing a localized search about the projected point. This helps to limit finding the closest point to a patch of the NURBS surface whose size will be proportional to the size of a grid cell.

3.3.2 Velocity Extraction from a Height Field

As an early test to determine how the height field could drive a physical simulation, only extracted pressures based on depth were used. However, it turned out

that pressures alone would only push the fluid from high areas of pressure to low areas which resulted in fluid that would just “flop in place”. This was undesirable because it wouldn't match the momentum of the incoming fluid flow from the height field. Instead, velocities were needed. It turns out that supplying a velocity field would be advantageous for a couple other reasons as well:

1. The velocities allow us to determine accurate pressures for the statistical domain by solving for pressure in eq. (3.14). These pressures can subsequently be used to provide an accurate pressure gradient in the blend domain.
2. The velocities can be used as the primary blending attribute in the blend domain, so there is be no need to try merging separate level sets from each domain. Only a single level set would be required that represented all the domains allowing for a desirable smooth surface.

A method to extrapolate velocities was determined by initially making the observation that level sets are advected in a velocity field according to the following equation:

$$\frac{\partial \phi}{\partial t} + \nabla \phi \cdot \vec{u} = 0 \tag{3.26}$$

Expanding this equation into three dimensions yields:

$$\frac{\partial \phi}{\partial t} + u\frac{\partial \phi}{\partial x} + v\frac{\partial \phi}{\partial y} + w\frac{\partial \phi}{\partial z} = 0 \tag{3.27}$$

By taking a simple Forward Euler step for the time derivative and a 2nd order accurate central difference for the spatial derivatives, we have:

$$\frac{\phi_{i,j,k}^{n+1}-\phi_{i,j,k}^{n}}{\Delta t}+u_{i,j,k}^{n}\frac{\phi_{i+1,j,k}^{n}-\phi_{i-1,j,k}^{n}}{2\Delta x}+v_{i,j,k}^{n}\frac{\phi_{i,j+1,k}^{n}-\phi_{i,j-1,k}^{n}}{2\Delta y}+w_{i,j,k}^{n}\frac{\phi_{i,j,k+1}^{n}-\phi_{i,j,k-1}^{n}}{2\Delta z}=0 \quad (3.28)$$

Of course, more accurate finite difference approaches could be used, but for the purpose of generating cartoon-looking fluid, this equation suffices. Additionally, this equation is convenient for it avoids the need to solve a large system of equations.

Note in eq. (3.28) that a level set from the statistical domain is needed at the current time n and at time $n+1$ for an appropriate time derivative. This is not difficult for us to do since the FFT height field representation is temporally independent. In fact, other than at time 0, only the level set at time $n+1$ will need to be generated per time step since we can reuse the level set from the previous iteration.

Now that we have level sets defined at time n and at time $n+1$, we have all that's needed to compute the derivatives. However, we still need a method to solve for the velocity components u, v, and w. We could do something similar to eq. (3.14) in the stable fluid method and generate a large system of equations given the incompressibility condition, but this would be a very expensive operation and wouldn't necessarily give pleasing velocities. Instead, the v component of the velocity can be solved directly since we have a temporally changing height field:

$$v = \frac{\partial h}{dt} = \frac{h(\mathbf{x}, n+1) - h(\mathbf{x}, n)}{\Delta t} \tag{3.29}$$

Note that we already have the heights defined at time n and $n+1$, so this is very straight forward to solve.

Since we are not concerned with physically accurate velocities, the other velocity components u and w can be set arbitrarily as long as they, combined with v, accurately satisfy the level set advection equation (3.27). All we need is a velocity field that will advect the level set to match that from the statistical domain. To keep things somewhat consistent we arbitrarily chose the $x:z$ ratio of the surface normal of the sample point in question as a second equation to help solve for u and w.

However, this approach doesn't solve for the velocities everywhere in the discretized environment. It only determines the velocities at the surface. To extrapolate the surface velocities into the air, the velocity extrapolation technique described in the PLS method [5] is used. Velocities inside the fluid are defined by simply decreasing the surface velocity exponentially with depth as was done by Yuksel, et al. [32].

3.3.3 Stable Fluid Modifications

Now that we have reasonable velocities defined in the statistical domain, we can proceed to solve the stable fluid method for both the blend and physical domains at the same time, but first, a couple initialization steps are required: initializing velocities in both domains and solving the pressures in the statistical domain for an accurate pressure gradient.

Velocities in the blend and physical domains are defined by periodically repeating the extracted velocities from the statistical domain in the exact same way the level sets are initialized. Pressures in the statistical domain, on the other hand, are found by using the extracted velocities as input for eq. (3.14) and solving for pressure. Now we can proceed to solve the Navier-Stokes equations in the usual manner except that we will use these pressures when computing the pressure gradient where the blend domain meets the statistical domain. This requires a slight modification when setting up the system of linear equations used to solve for pressure in eq. (3.14).

Finally, we linearly blend the extracted velocities from the statistical domain with the divergence-free velocity field found by solving the Navier-Stokes equations in the blend domain and subsequently update the level set. This allows for a relatively smooth transition between the statistical and physical domains and provides a way for a height field to drive a physical simulation.

In summary, here's our simulation routine that's run at every time step to solve the Navier-Stokes equations and blend the domains:

$$
\begin{aligned}
\phi^{n}_{height} &= \phi^{n+1}_{height} \\
\phi^{n+1}_{height} &= convertToLevelSet(h(n+1)) \\
\vec{u}_{height} &= extractHeightVelocities(\phi^{n}_{height}, \phi^{n+1}_{height}) \\
\nabla^2 p_{height} &= \Delta t \frac{1}{\rho} \nabla \cdot \vec{u}_{height} \\
\mathbf{W}^0 &= \vec{u}^n \\
\mathbf{W}^1 &= advect(\mathbf{W}^0, -\Delta t) \\
\mathbf{W}^2 &= \mathbf{W}^1 + \Delta t \mathbf{f} \\
\nabla^2 p &= \Delta t \frac{1}{\rho} \nabla \cdot \mathbf{W}^2 \text{ using } p_{height} \text{ at the appropriate boundary} \\
\mathbf{W}^3 &= \mathbf{W}^2 - \Delta t \frac{1}{\rho} \nabla p \\
\vec{u}^{n+1} &= blend(\vec{u}_{height}, \mathbf{W}^3) \\
\vec{u}^{n+1} &= extrapolateVelocities(\vec{u}^{n+1}) \\
\phi^{n+1} &= advect(\phi^n, \Delta t)
\end{aligned}
\tag{3.30}
$$

3.4 Rendering

There are numerous ways to do non-photorealistic renderings of 3D objects from toon/cell shaders to painterly renderings [24]. In order to achieve a classic Disney style look such as that from *Alice in Wonderland* [10], I paid close attention to the color palette and element movement on the water. Two primary concepts were developed to achieve certain elemental looks: edge line drawings and toon-like reflections.

3.4.1 Edge Drawings

In order to draw edges, an edge-detection method was needed. This is done by a post process that uses a Sobel gradient filter on a depth image created at render time. These Sobel filters in both the x and y directions are:

$$Sobel_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} \quad \text{and} \quad Sobel_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Together, these filters provide a gradient vector in the direction of largest increasing depth.

When the length of the gradient vector at a pixel exceeds a user-defined threshold, an edge is determined and will be colored a user-specified color. Otherwise the pixel will be set to be fully transparent for easy compositing. To reduce aliasing artifacts, the Sobel filters were extended to be 7x7 filters, allowing for thicker edges that are blurred by a small Gaussian filter.

3.4.2 Reflection Elements

Upon careful examination of the *Alice in Wonderland* "Through the Keyhole in a Bottle" sequence [10], some stylized elements are present on the surface of the fluid that mimic realistic phenomena. One such element appears to be a stylized reflection of the stormy sky which appears to be only present on the front side of the wave face. As a preliminary way of achieving the same effect, a simple expression based on the normal

and velocity at a certain point was established. A camera-facing normal with a positive v velocity greater than some threshold resulted in a toony reflection located on the front side of a wave. An example of a cartoon render achieved can be seen in Figure 7.

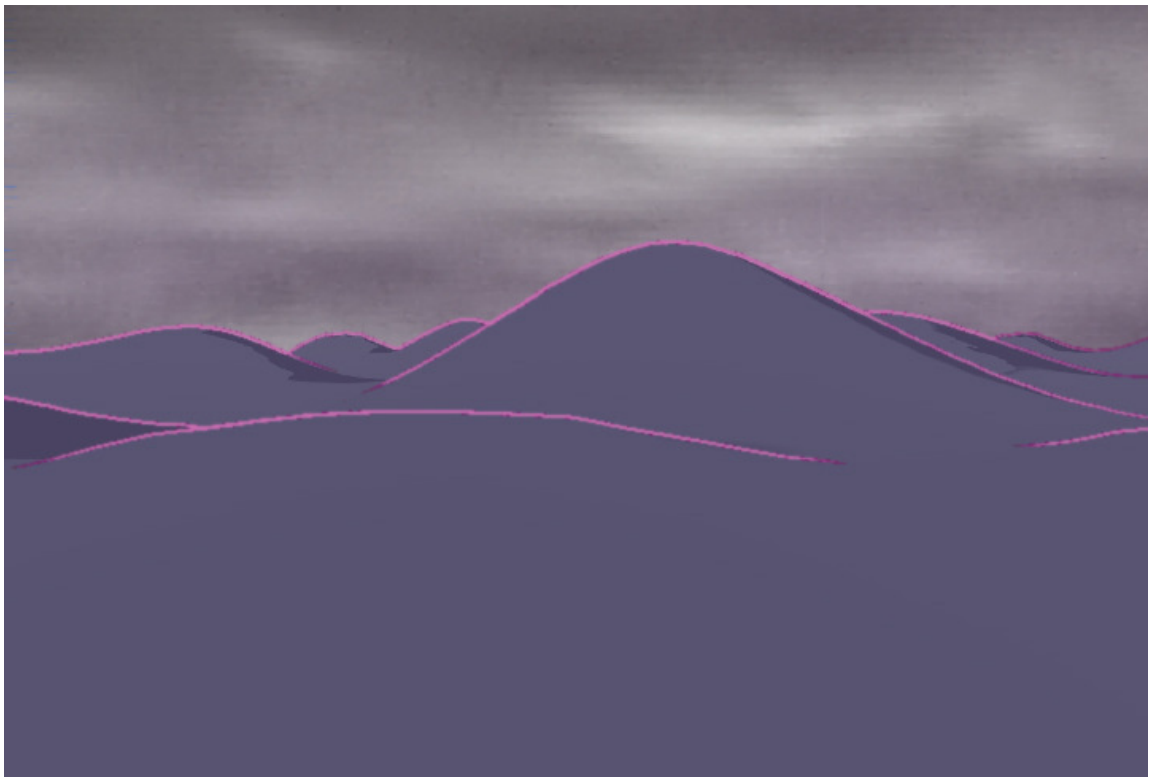

Fig. 7. Cartoon Render.

4 RESULTS

The methodology described was implemented with mixed results. In order to clearly describe the results, this section will continue by discussing each of the primary methodology objectives in turn starting with wave generation, followed by results from our stable fluid solver, continuing with the blend domain and its effectiveness, and concluding with rendering.

4.1 Wave Generation

It was determined after careful examination that using a statistical approach to generate waves via an FFT representation was ideal to solve the problem of creating waves that could be stylistically controlled. The statistical model proved to be a useful means by which a user could model a fluid surface and still exhibit believable motion. In fact, the method would also allow for keyframe control of realistic fluids if the user so chose leaving them with total creative freedom. A realistic render of a fluid surface using our height field method can be seen in Figure 8. Additionally, the FFT-based approach turned out to be quite useful as it allowed for real-time, or near real-time (depending on sample size) wave animations providing the user with quick feedback.

Fig. 8. Realistic Ocean Surface. Rendered using a 512x512 statistical domain.

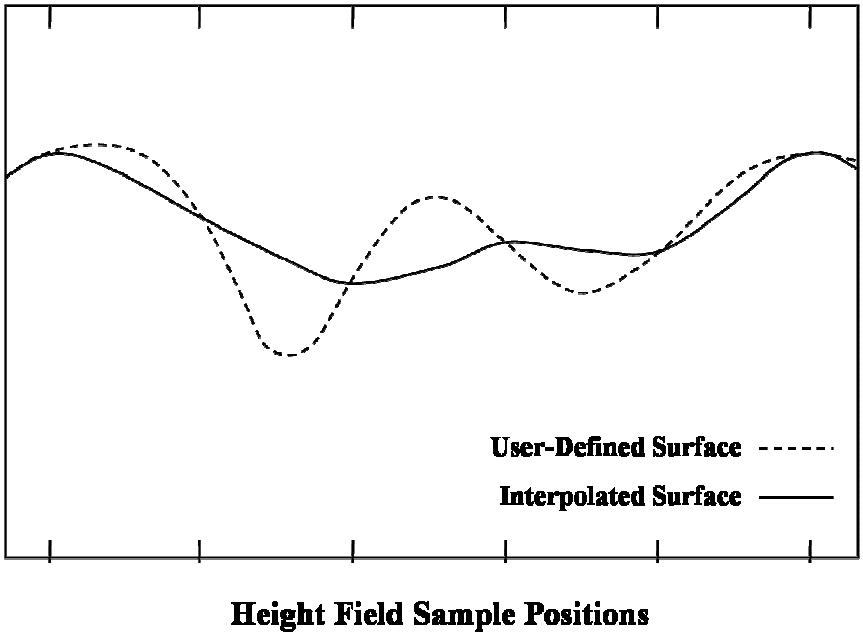

Fig. 9. Lost Detail due to Inadequate Sampling.

Modeling of the fluid surface was provided by traditional NURBS modeling techniques which turned out to be quite useful for the purpose of cartoon-like fluid, which in many cases is very smooth and rounded. In fact, due to the simplicity of converting a height field into a Fourier-domain representation, any traditional 3D surface representation could be used (polygons, sub-division surfaces, etc.) as long as they can be easily sampled at discrete points.

Using the FFT-based approach came with its restrictions, however, the biggest of which is its unintuitiveness. For example, in order to correctly apply an FFT to a modeled surface, it needs to be sampled at discrete, evenly-spaced points, and the number of samples must be a power of 2. If a user modeled a fluid surface with fine detail that wasn't appropriately sampled, that detail would be lost as is shown in Figure 9. Of course, the user could increase the number of samples to capture the detail, but that might come at a cost because it needs to be understood that each sample represents a wave of a particular wavelength in the Fourier domain. For instance, if a surface is sampled by an 8x8 uniform grid, its transformation into the frequency domain will result in 64 different waves. If a 64x64 uniform grid is used, 4096 different waves will be generated. Figure 10 shows how the addition of samples creates new, higher frequency waves. The final number of waves can be controlled somewhat by providing custom spectral functions. For instance, using equation (3.4) will eliminate any waves that face against the wind, but providing spectral functions is certainly not an intuitive way to control the final look of a fluid.

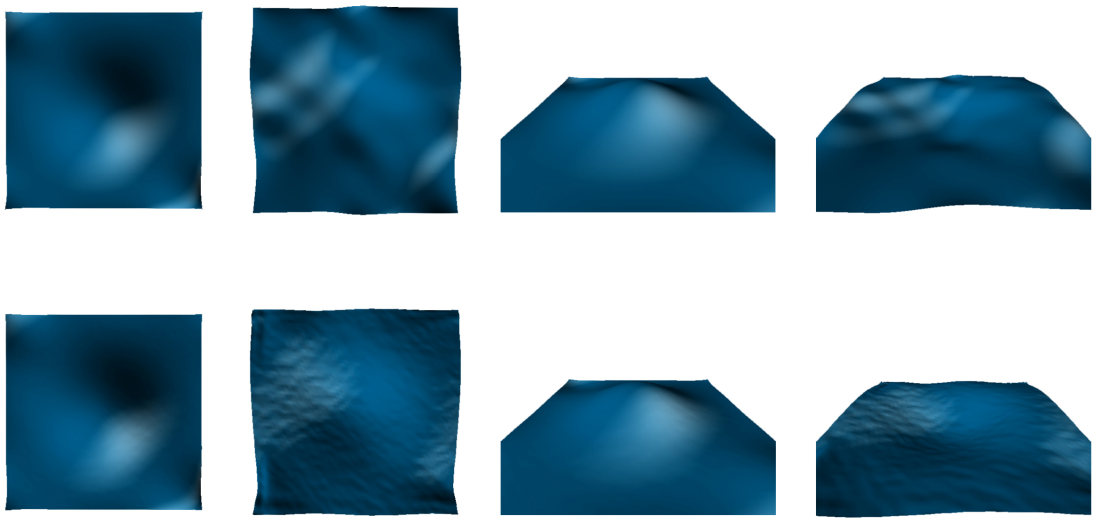

Fig. 10. Sample Number Comparison. The top row uses an 8x8 grid of uniform samples. The bottom row is 64x64. Notice how much more detail is added by simply increasing the number of samples.

4.2 Dynamic Fluid

To solve the 3D Navier-Stokes equations, we employed the Stable Fluid method introduced by Stam [28]. To track the evolution of the fluid surface, the PLS method by Enright, et al. [4] was used. Both methods are quite successful at providing a means to simulate and render realistic fluid and are common in modern production environments. Our implementation of them was equally successful as can be seen in Figure 11 and in the supplemental material stablefluid.mov (available for download).

However, these methods provide challenges particularly for large-scale simulations because of their CPU, memory, and disk-space requirements. For instance, using our implementation to simulate a drop of water falling into a pool as seen in Figure

11 took approximately 1.5 minutes per frame to simulate on a 2 GHz Athlon processor and used 500 MB of RAM using a 50x50x100 resolute MAC grid - all this for a relatively simple simulation. More complex simulations would require even more resources. There does exist in the research literature a handful of different optimization techniques such as those described in [21], [15], and [16], but they weren't utilized in our implementation as achieving optimal simulation speed was outside the scope of this thesis.

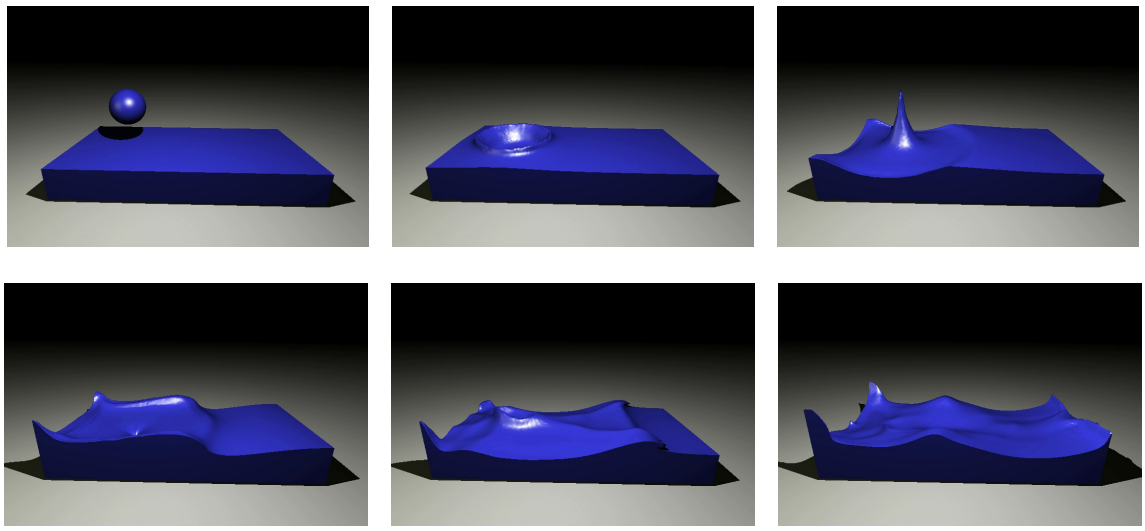

Fig. 11. Dynamic Fluid Results. Example of a ball of water dropping into a pool using the Stable Fluid and PLS methods.

4.3 Blending of the Statistical and Physical Domains

The results of the blend domain in our simulation were mixed primarily due to the velocity extraction technique. The method was successful at providing a way to

smoothly blend between a height field and a physically-solved fluid surface allowing for the height field to drive dynamic effects such as breaking waves on a beach, but noticeable bump artifacts were present due to two primary factors:

1. The x:z ratio of the level set gradient used to help solve for the u & w components of the velocity wasn't always stable. For instance, as the z component of the gradient approaches 0, the ratio becomes exponentially larger, resulting in potentially extreme velocity components.
2. The linearly interpolated blend of the velocities from the statistical and physical domains doesn't necessarily result in a smooth velocity field. Smooth velocities are necessary for an unconditionally smooth fluid surface after advection.

The unsmooth velocity field only intensified the artifacts in a PLS simulation since particles used to correct the level set could be several grid cells away from the surface where the velocities aren't as well defined. By removing the particle correction step, the fluid surface was much smoother, but artifacts were still noticeable. Frames from a level set-only and a PLS simulation can be seen in Figures 12 and 13 respectively along with a single frame comparison of the two in Figure 14. The supplemental material blendcomparison.mov (available for download) compares the fully simulated fluid from both techniques as well.

The size of the blend domain also had an impact on the effectiveness of the blend particularly for reflected waves off a wall or beach. Due to the linearly interpolated

velocities, these reflection waves disappear as they approach the statistical domain. This is much more noticeable to the naked eye when the blend domain is small in size. With a larger domain, it looks rather natural. In fact, experiments verified that a blend domain at least the same size as the statistical domain was reasonable.

Of course, using a blend domain that's at least the same size and resolution as the statistical domain results in a much larger total simulation area, drastically increasing the amount of resources needed. On a 60x60x180 resolution simulation domain, it took approximately 3.5 minutes a frame to simulate on a 3GHz athlon processor and used approximately 1GB of memory, not to mention that caching the entire level set on disk in ASCII format for future rendering took approximately 100MB of disk space. This could be made much more efficient by employing several optimization techniques, but that is left for future work.

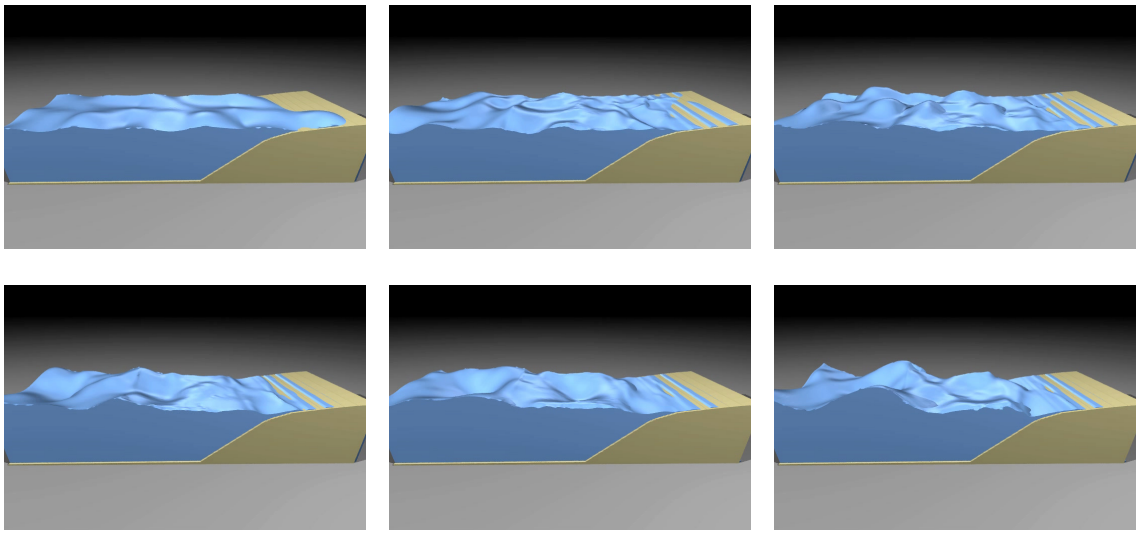

Fig. 12. Level Set-only Simulation Results.

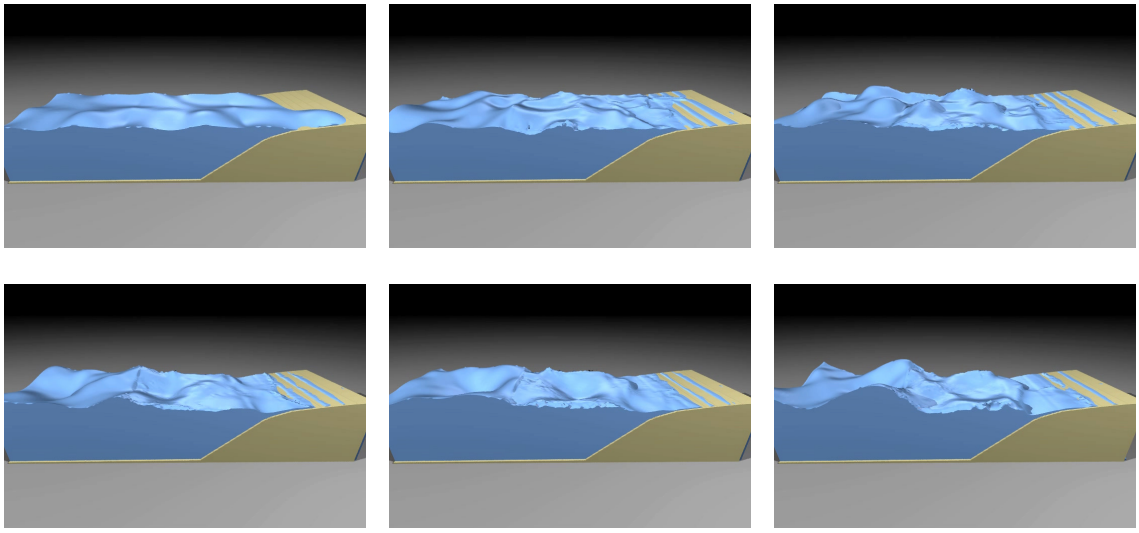

Fig. 13. PLS Simulation Results. Uses the same initial conditions as Figure 12.

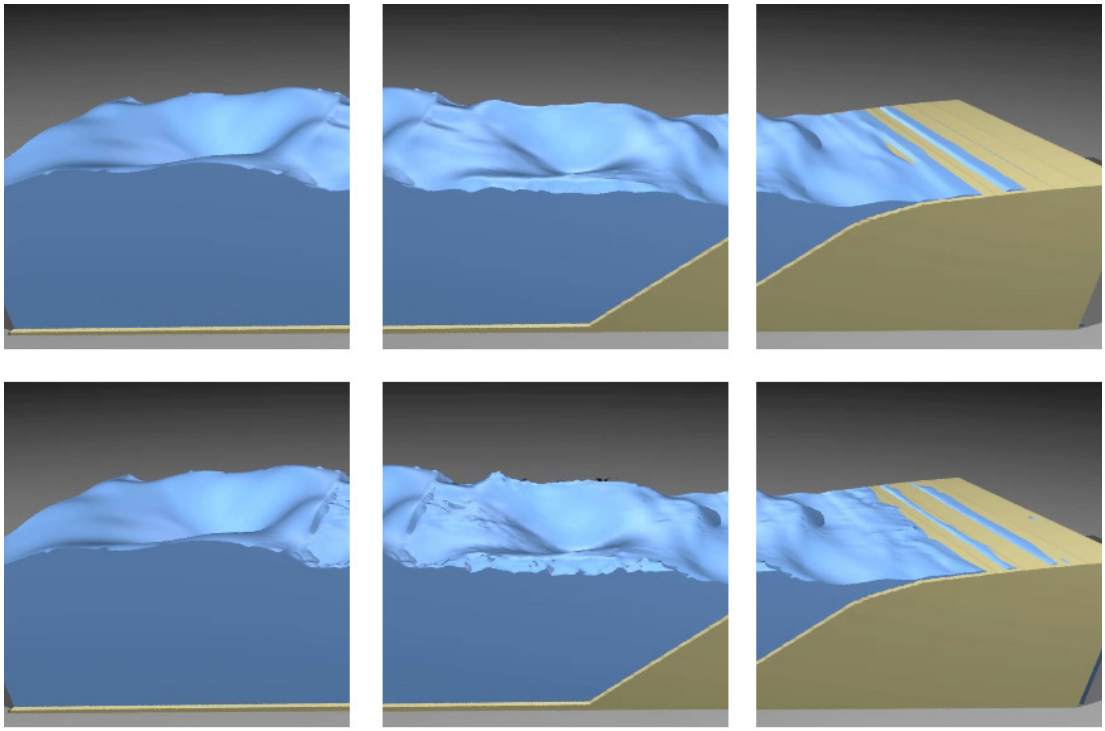

Fig. 14. Comparison between the Level Set-only and PLS Simulations in the Different Domains. The top row is the level-set only simulation and the bottom is the PLS.

4.4 Rendering

Since the fluid is represented by a fully-renderable implicit surface, any number of rendering techniques could be used from the photoreal to non-photoreal. It all depends on the intended look, and our level set is fully capable of providing a surface representation that can be used in an infinite number of ways.

As an experiment to create a stylized fluid, we modeled, simulated, and rendered a fluid in a manner influenced by the "Through the Keyhole in a Bottle" sequence from *Alice in Wonderland* [10]. The level set was rendered using a flat-shading technique without lighting and made use of a depth map to produce some subtle depth-of-field effects. The depth map was additionally used to create edge lines in a post process that used a traditional edge-finding image filter. Results of our cartoon simulation can be seen in Figure 15 and in the supplemental material cartoonfluid.mov (available for download).

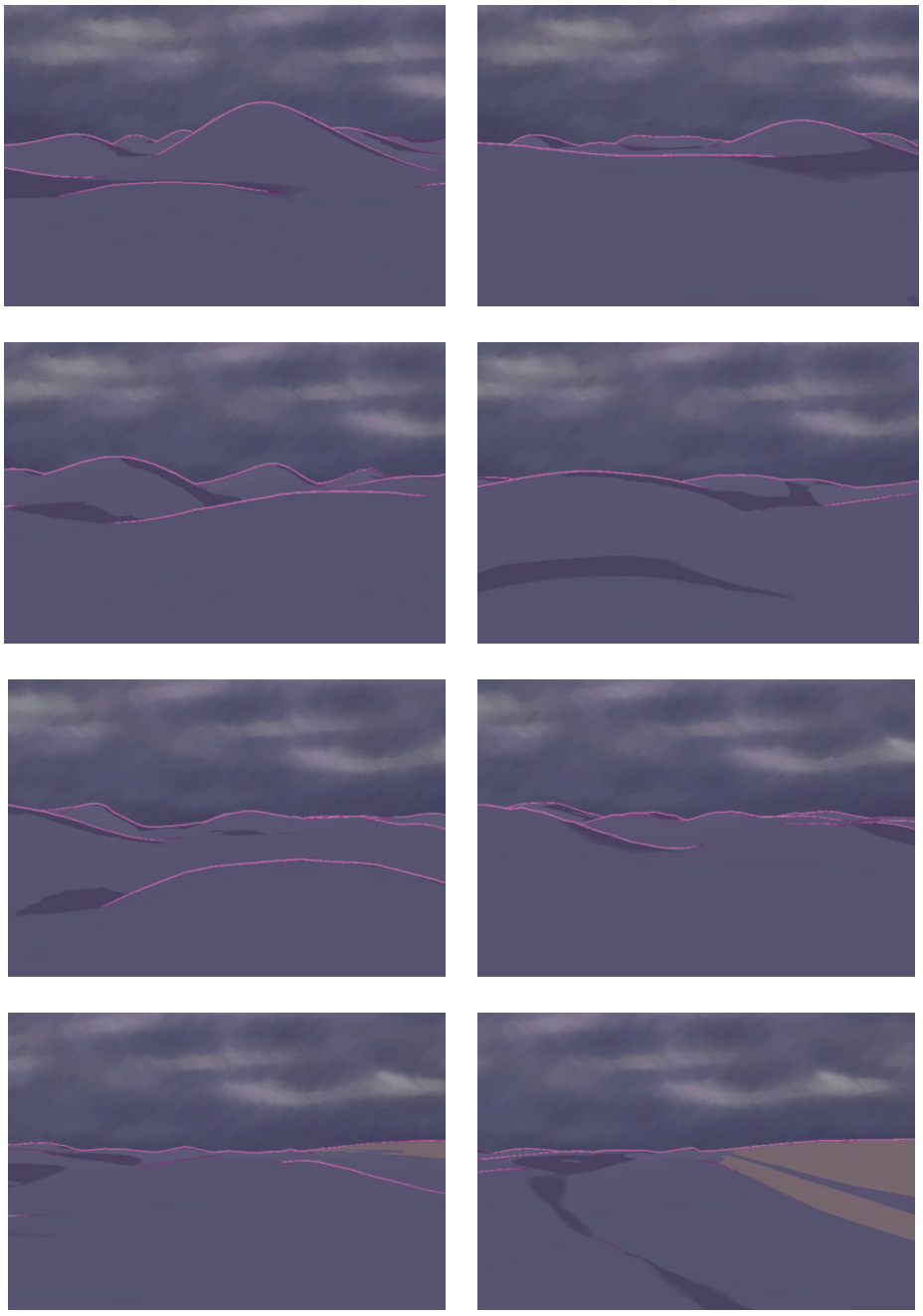

Fig. 15. Cartoon-rendered Simulation Results.

5 CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

In order to provide a mechanism to create stylized, yet believable fluids that make use of common fluid simulation techniques, this thesis successfully employed a new simulation technique that combines the commonly used FFT-based height field approach for fully developed seas and the Stable Fluid method for dynamic effects via a blend domain. Stylization is primarily achieved by keyframing wave amplitudes defined and animated in the Fourier domain that are derived from a modeled fluid surface. By linearly interpolating velocities extracted from this animated surface with velocities resulting from solving the 3D Navier-Stokes equations in a blend domain, the stylized surface can additionally drive dynamics such as breaking waves while still maintaining cohesiveness.

Although applying keyframes on frequency-based wave amplitudes might be somewhat limiting and unintuitive to a novice user, it is an effective way by which the look of a fluid surface can be stylized and yet exhibit natural fluid motion. Additionally, despite its artifacts, the current implementation of the blend domain has proven to be an interesting way for a height field to drive a dynamic simulation, carrying with it potential for future research.

By applying both techniques and making use of a non-photorealistic level-set rendering algorithm, it has been shown that stylizing fluids in a cartoon manner is certainly possible. In fact, due to the relative simplicity of combining the common

simulation techniques used, such a methodology could easily be incorporated into modern production fluid pipelines.

5.2 Future Work

This thesis carries with it a lot of interesting future work, both in improving its limitations and in new research. This sub-section will proceed by discussing a few ideas for each.

5.2.1 Limitation Improvements

Limitations introduced by using the FFT-based height field approach could be improved by using an alternate method to solve the discrete fourier transform such as an NDFT (nonequidistant discrete fourier transform) method described by [3]. This would be slower, but it could potentially allow the placement of discrete sample points anywhere on the modeled fluid surface to capture detail that might have been lost with evenly distributed samples. It would also be interesting to see how each discrete sample would then be interpreted in the frequency domain. In other words, could a user control the total number of waves used to build and animate the surface?

Another limitation of the FFT-based approach is that it results in a very smooth height field which is even further smoothed by fitting it with a NURBS surface. If a peaked surface is the intended look, this approach wouldn't work. Our method could be extended to include peaked surfaces by employing an additional FFT-based displacement vector field as described in [30], but such a peaked surface might still be

difficult to capture in an inherently smooth level set. Nevertheless, it might be worthwhile exploring whether this displacement vector field can be used as a displacement map on the level set instead.

Of course our approach can be used for realistic water too, although some additional limitations would need to be overcome in this case. For instance, for a really detailed fluid surface, the number of samples used in the height field would need to be quite large, e.g. 2048x2048 samples = 4,194,304 total waves. To preserve that kind of detail in a level set would require a very fine grid and therefore be very expensive. Instead, one might be able to use only the large wave vectors from those samples and preserve the small waves in a displacement map to be applied to the level set. This way the large waves can still be used to drive the physical simulation, and the fine detail can be preserved using a sequence of displacement maps.

Our Stable Fluid and Particle Level Set methods can be optimized by making use of several efficiency techniques such as octree data structures [21], minimizing the level set to only be defined and advected near the surface using Run-Length Encoded grids [15], and using tall grid cells inside the fluid where not as much detail is necessarily needed [16]. Implementing such techniques would drastically reduce the complexity and resources needed for our method. Additionally, effects like spray, mist, foam, and bubbles can also be created by making use of the escaped particles from the PLS method.

5.2.2 Future Research

Future research might include finding ways to reduce the artifacts in the blend domain. There potentially exists a few different ways to do this. One might be to apply a smoothing operator on the level set to smooth out any high-frequency noise, but this might result in the loss of desired detail. Another possibility might be to find better ways to extract the velocities from the height field. One such way would be to somehow find the true direction of propagation of the fluid surface at any point. This would result in better velocities for the u and w components rather than arbitrarily picking them. Additionally, applying some kind of "smooth" restriction to the velocities might need to be explored as well to reduce artifacts altogether.

Alternatively, rather than trying to blend velocities in the blend domain, future research might explore blending two level sets instead: the statistical domain level set and the level set resulting from a PLS advection in the physical domain. Since level sets are implicit functions, blending of the two could be easily done. In fact, such an approach might allow for wave reflections to flow from the physical domain into the statistical domain therefore providing a potential way to generate a height field from a physical simulation.

Our technique could be utilized in other interesting fluid applications as well such as a boat or object floating in a large body of water. The majority of the water could be animated and rendered using a user-controlled height field technique, but the fluid directly surrounding the floating/colliding object could be physically simulated and then blended with the height field. This might drastically reduce the size of the physical

domain that might have otherwise been necessary resulting in much faster simulations that use far less resources. Moreover, such a concept leads to the idea of a dynamically changing simulation domain. For instance, if an object is thrown into a body of water, the area directly surrounding the object at impact could be converted into a dynamic simulation and blended with the rest of the fluid without the need of an unnecessarily large physical simulation domain.

REFERENCES

[1] R. Bridson and M. Müller-Fischer, "Fluid Simulation: SIGGRAPH 2007 Course Notes," *ACM SIGGRAPH 2007 Courses,* course 31, pp. 1-81, 2007.

[2] P.W. Cleary, S.H. Pyo, M. Prakash and B.K. Koo, "Bubbling and Frothing Liquids," *ACM Trans. Graphics*, vol. 26, no. 3, article no. 97, July 2007.

[3] A. Dutt and V. Rokhlin, "Fast Fourier Transforms for Nonequispaced Data," *SIAM J. Scientific Computing,* vol.14, no.6, pp.1368-1393, Nov. 1993.

[4] D. Enright, R. Fedkiw, J. Ferziger and I. Mitchell, "A Hybrid Particle Level Set Method for Improved Interface Capturing," *J. Computational Physics*, vol. 183, no. 1, pp 83-116, Nov. 2002.

[5] D. Enright, S. Marschner and R. Fedkiw, "Animation and Rendering of Complex Water Surfaces," *ACM Trans. Graphics*, vol. 21, no. 3, pp. 736-744, July 2002.

[6] R. Fattal and D. Lischinski, "Target-driven Smoke Animation," *ACM Trans. Graphics,* vol. 23, no. 3, pp. 441-448, Aug. 2004.

[7] N. Foster and R. Fedkiw, "Practical Animation of Liquids," *Proc. ACM SIGGRAPH '01*, pp. 23-30, 2001.

[8] N. Foster and D. Metaxas, "Realistic Animation of Liquids," *Graphical Models and Image Processing*, vol. 58, no. 5, pp. 471-483, Sept. 1996.

[9] A. Fournier and W.T. Reeves, "A Simple Model of Ocean Waves," *Proc. ACM SIGGRAPH '86*, pp. 75-84, 1986.

[10] C. Geronimi, W. Jackson and H. Luske Directors, *Alice in Wonderland*, DVD, Burbank, C.A.: Walt Disney Productions, 1951.

[11] F.J. Gerstner, *Theory of Waves*, Berkeley: Univ. of California, Inst. Engineering Research, 1952.

[12] S.T. Greenwood and D.H. House, "Better with Bubbles: Enhancing the Visual Realism of Simulated Fluid," *Proc. 2004 ACM SIGGRAPH/Eurographics Symp. Computer Animation*, pp. 287-296, 2004.

[13] E. Guendelman, A. Selle, F. Losasso and R. Fedkiw, "Coupling Water and Smoke to Thin Deformable and Rigid Shells," *ACM Trans. Graphics*, vol. 24, no. 3, pp. 973-981, July 2005.

[14] F. Harlow and J. Welch, "Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with a Free Surface," *The Physics of Fluids*, vol. 8, pp. 2182-2189, 1965.

[15] B. Houston, M. Wiebe and C. Batty, "RLE Sparse Level Sets," *ACM SIGGRAPH 2004 Sketches,* p. 137, 2004.

[16] G. Irving, E. Guendelman, F. Losasso and R. Fedkiw, "Efficient Simulation of Large Bodies of Water by Coupling Two and Three Dimensional Techniques," *ACM Trans. Graphics,* vol. 25, no. 3, pp. 805-811, July 2006.

[17] O. Johnston and F. Thomas, *The Illusion of Life: Disney Animation*, p. 257. New York: Walt Disney Productions, 1981.

[18] B. Kim, Y. Liu, I. Llamas, X. Jiao and J. Rossignac, "Simulation of Bubbles in Foam with the Volume Control Method," *ACM Trans. Graphics*, vol. 26, no. 3, article no. 98, July 2007.

[19] J. Kim, D. Cha, B. Chang, B. Koo and I. Ihm, "Practical Animation of Turbulent Splashing Water," *Proc. 2006 ACM SIGGRAPH/Eurographics Symp. Computer Animation*, pp 335-344, 2006

[20] P. Lavoie, "The NURBS++ Package," May 2002; http://libnurbs.sourceforge.net/index.shtml.

[21] F. Losasso, F. Gibou and R. Fedkiw, "Simulating Water and Smoke with an Octree Data Structure," *ACM Trans. Graphics,* vol. 23, no. 3, pp. 457-462, Aug. 2004.

[22] G.A. Matsin, P.A. Watterberg and J.F. Mareda, "Fourier Synthesis of Ocean Scenes," *IEEE Computer Graphics and Applications*, vol. 7, no. 3, pp. 16-23, Mar. 1987.

[23] A. McNamara, A. Treuille, Z. Popovic and J. Stam, "Fluid Control Using the Adjoint Method," *ACM Trans. Graphics*, vol. 23, no. 3, pp. 449-456, Aug. 2004.

[24] B.J. Meier, "Painterly Rendering for Animation," *Proc. SIGGRAPH '96,* pp. 477-484, 1996.

[25] V. Mihalef, D. Metaxas and M. Sussman, "Animation and Control of Breaking Waves," *Proc. 2004 ACM SIGGRAPH/Eurographics Symp. Computer Animation*, pp. 315-324, 2004.

[26] L. Piegl and W. Tiller, *The NURBS Book*, London: Springer-Verlag, 1995.

[27] J.A. Sethian, “Fast Marching Methods,” *SIAM Review*, vol. 41, no. 2, pp. 199-235, 1999.

[28] J. Stam, “Stable Fluids,” *Proc. SIGGRAPH ’99*, pp. 121-128, 1999.

[29] M. Sussman and E.G. Puckett, “A Coupled Level Set and Volume-of-Fluid Method for Computing 3D and Axisymmetric Incompressible Two-Phase Flows,” *J. Computational Physics*, vol. 162, no. 2, pp. 301-337, Aug. 2000.

[30] J. Tessendorf, “Simulating Ocean Water,” *ACM SIGGRAPH 2004 Courses*, course 31, 2004.

[31] A. Treuille, A. McNamara, Z. Popovic and J. Stam, “Keyframe Control of Smoke Simulations,” *ACM Trans. Graphics,* vol. 22, no. 3, pp. 716-723, July 2003.

[32] C. Yuksel, D.H. House and J. Keyser, “Wave Particles,” *ACM Trans. Graphics*, vol. 26, no. 3, article no. 99, July 2007.

VITA

Name:	Christopher Wayne Root
Address:	Industrial, Light, & Magic P.O. Box 29919, San Francisco, CA 94129
Email Address:	cwr@viz.tamu.edu
Education:	B.S., Computer Science, University of Minnesota (Twin Cities), 2002
Employment:	Software Engineer/Consultant, Syntegra, USA, 2002-2003 Technical Director Intern, Pixar Animation Studios, 2004 Teaching/Research Assistant, Texas A&M University, 2005-2006 Assistant Technical Director, Industrial Light & Magic, 2006-present