Efficient Modeling and Rendering of Turbulent Water over Natural Terrain

Nathan Holmberg* and Burkhard C. Wünsche†
Graphics Group, Department of Computer Science
University of Auckland, Auckland, New Zealand

Abstract

Water phenomena are some of the most visually spectacular effects found in nature. This paper presents an efficient hybrid method to model turbulent water such as fast flowing rivers and waterfalls with the intent that the model can be used as part of a larger environment or scene. The model presented uses hydrostatic theory to incorporate a 2D height field and a particle system to model respectively the main volume and spray of turbulent water. The user is able to submit any environment formed from spheres and panels making the solution very flexible and adaptable.

A smooth representation of the water surface is obtained by fitting a uniform B-Spline surface to the height field. Foam, spray and other turbulent effects are represented by particles which are rendered as spheres or billboards. Our results show that the model provides a nearly realistic simulation of turbulent water and for simple scenes nearly interactive speeds are possible which compares favorably with alternative techniques. For non-interactive applications ray tracing can be used to obtain higher quality results.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation;

Keywords: physically-based modeling, turbulent water, water simulation

1 Introduction

The complexity and power of water flow in nature is both impressive and beautiful. It is also a part of our everyday lives and so well known to the human perception that unrealistic motion is easily discernible. It is no surprise therefore that much effort has been applied in computer graphics to try to capture water in a convincing way. This is far from trivial as most phenomena can be attributed to complex rapidly changing molecular interactions, and as such are beyond the modeling power of today's computers. Instead computer graphics researchers try to produce models with underlying physical principles that create, as accurately as possible, large scale approximations of these local interactions.

*e-mail: nhol021@ec.auckland.ac.nz
†e-mail: burkhard@cs.auckland.ac.nz

Research on the simulation of the behavior of water has progressed from models which were only able to represent certain effects, such as the motion of deep water waves to the exclusion of all else, to more general models with a firmer basis in hydrodynamics capable of realistic motion that follows expected behavior in a range of situations. These more recent models allow animators to specify environments and start conditions and the model will do the rest.

The purpose of this work is to build a general model with a focus on the natural movement of rivers, rapids and waterfalls. As such turbulence, spray, and the like should be accounted for while not being treated as special cases. While not yet being complete and fully realistic, our model has been able to produce results that show its potential.

2 Previous Work

The earliest work on modeling water in computer graphics used mathematical models and explicit functions to simulate surface behavior. Some, while obeying physical equations, were very inflexible outside their intended scope such as the modeling of deep sea waves by Schachter [Schachter 1980] while others tried to model surface phenomena in 2D velocity fields [Neyret and Praizelin 2001].

Later work introduced rudimentary particle systems. Initially particle interaction was ignored and particles simply bounced around the environment [Reeves 1983; Sims 1990]. This produced reasonable effects for waterfalls and other instances where particle movement had enough energy to break the molecular level bonds as needed. In particle systems the equations for interaction can be complex and computationally expensive as each particle may be affected by every other resulting in a computational complexity of $O(n^2)$ which is unacceptably slow when dealing with hundreds of thousands of particles. However, without inter-particle forces volumes of water can not be modeled well and the usually identifiable effects of adhesion and cohesion are missing. Some of the work at modeling these effects includes Miller and Pearce's [Miller and Pearce 1989] connected particle system which used forces to imitate soft collisions based on the difference in particle's positions. This works well for viscous fluids, as intended by the authors, but is unsuitable for the number of particles or scale necessary to model large bodies of water.

Kass and Miller [Kass and Miller 1990] introduced column systems which use simplified flow equations, based on hydrostatics, between columns and treat the water volume as a 2D height field. This implicitly allows for the modeling of surface phenomena such as waves and can efficiently model large bodies of water. O'Brien and Hodgins [O'Brien and Hodgins 1995] extended this model by incorporating the interaction with external objects and splashes. However, a column's height is still represented by one height variable meaning that vertical isotropy is assumed and only very simple (flat) environments can be used. Mould and Yang [Mould and Yang 1997] furthered this work by dividing each column into a user defined number of cells, relaxing the assumption of vertical isotropy. Another improvement was made by allowing for more complex en-

vironments.

Even with the use of cells, however, column systems are still only two and a half dimensional models at best and many effects such as the curling of waves cannot be reproduced. There has also been some work at applying the 2D version of the Navier-Stokes equations to column systems [Chen and Lobo 1995].

The most recently developed simulation methods use the full Navier-Stokes equations to compute motion in a 3D grid of voxels, examples of which can be found in Enright et al. [Enright et al. 2002] and Foster and Fedwik [Foster and Fedkiw 2001]. The solution is very expensive but also creates the most realistic effects. It is obtained by calculating the velocity of the fluid flow out of the 6 faces of each voxel using the pressure, temperature etc. of each voxel. By calculating which voxels contain water, objects, or air, extremely realistic simulations are obtained for complex environments. This method has been used in movies such as Shrek and its difficulty is perhaps illustrated in Jeffrey Katzenberg's statement that the pouring of milk into a glass was the hardest shot in the movie [Foster and Fedkiw 2001; Enright et al. 2002]. This approach does have its limitations of course with perhaps the largest being the loss of information within cells, meaning that the grid size plays an overly important role in the accuracy of the model.

3 A Hybrid Model for Turbulent Water

Initial tests with a particle only model showed that the computation time for such a system is prohibitive for large bodies of water. Instead we chose to use different systems for spray and volume to create a faster and more believable model.

3.1 Volume Model

Continuing the work of O'Brien & Hodges and Mould & Yang [O'Brien and Hodgins 1995; Mould and Yang 1997] the volume model, representing the main body of water in our model, is a column based system. Using columns holds the advantages of easy surface creation as the top of all columns are known and less flow calculations are needed so the system is less computationally intensive.

Our model divides the environment into equally sized squares which form the base of columns as in figure 1. All columns start with a user defined height which then varies over time dependent on the calculated flows. Source and sink columns are the only ones to retain their heights allowing for in and out flows to the system. Pipes are then created between each of the eight adjacent columns and each cell that could overlap during the course of the simulation. Pipes are also created between the cells of one column and the air above the adjacent columns. At this point the system is ready to begin simulation.

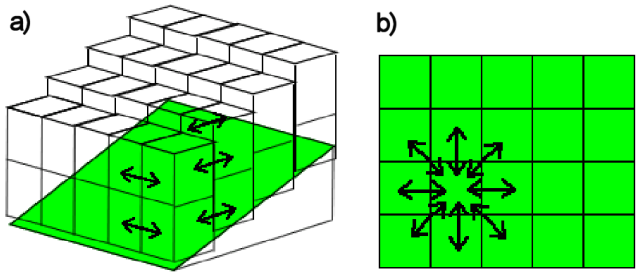

Figure 1: Columns generated for a sloped environment (a) with pipes between columns (b).

The underlying basis for the flow equations that are used in this simulation is the science of hydrostatics, which describes the pressure of fluids at rest. The equations related with this approach are simple, both to understand and to compute, and as a result are easy to implement.

For any column in the grid the hydrostatic pressure can be calculated from the equation

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

which is based on the work of Bernoulli and where Q is the total pressure, h the height of the column, g the acceleration due to gravity, ρ the density of the fluid, v the velocity of flow, p_0 the air pressure and E the pressure arising from external forces which together form the pressure energy term. In this case the height of the column is the height above some arbitrary point in the world so long as the same point is used for all columns.

Using the pressure differences between cells it is possible to calculate the acceleration and from that the flow that should occur between cells. The final equation for the flow velocity (η) through a pipe is

$$\eta = f\eta_0 + \Delta t\left(\frac{Q_{head} - Q_{tail}}{\rho l}\right)$$

where l is the pipe length, f is a friction coefficient (as suggested in [Mould and Yang 1997]), and η_0 is the flow in the previous time step. An interesting point to note is the lack of any viscosity parameter in this equation. This is because one of Bernoulli's assumptions is that the distances between points of measurement are so small that viscous losses are negligible. Instead the friction coefficient used allows energy to slowly escape from the system. While not physically justified this parameter serves as an ad-hoc method of including viscosity and we found that setting it to 0.995 gave good results in our examples.

Using the flow calculated for the pipe the volume V of water that should be moved through it is calculated by:

$$V = \Delta t \eta C$$

where C is the cross-sectional area of the pipe, or the amount of overlap between the cells. Because mass is to be conserved the volume removed from one column is the same as that added to the other. Care must also be taken not to allow a volume of less than zero to occur.

This system is very fast considering the mass of water that is being represented but there are several problems. Columns pose a problem as a representation because one of the classical characteristics of turbulence is that it is a three dimensional feature [Schachter 1980]. While using the "cells" given by Mould and Yang relaxes the assumption of vertical isotropy the model is not capable of simulating certain situations such as vortices. A second problem arises from the fact that turbulence is a feature of flow and not of the fluid "at rest". This means that while hydrostatics may be easy to use the equations generated for flow are incomplete and ignore many of the visible characteristics of water such as viscous shear stresses. This is perhaps the largest flaw in the model and something that lends itself to further research.

3.2 Spray Model

The spray model is used to model water as it breaks free of the main volume of water. There is no easy physical solution to when spray should be created and as such assumptions must be made instead. Earlier work with column systems were concerned mainly with generating splashes from hitting objects and as such used vertical velocity thresholds for generating spray. Because we also want to model waterfalls our assumptions use research about the heights

of wave crests before they become unstable. Thornton and Guza determined that this occurs when the wave height is 0.78 of the water depth [Thornton and Guza 1982]. This obviously allows waterfalls to form easily but also works for rapids as large flow velocities form "spikes" of water that while erroneous are then turned into spray due to their large heights.

The spray system begins its evolution when particles are generated. First the number of particles (or volume) needs to be determined. Using the formula to calculate flow through a weir [Badger and Banchero 1955] it is possible to determine the required flow rate ζ and hence the volume:

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

where B is the base length, H the height and g the acceleration due to gravity. The volume V to pass through in this time step is then given by

$$V = \zeta \Delta t$$

This determines the number of particles to be created as all those generated are of a user defined volume except the last which needs to be the remainder of volume to be moved as mass must be conserved and not created. Depending on the scale and resolution of the model being used this can be set to achieve the best looking results. The position of the particles is also easily determined and is set at a random position in the face that the particle is being generated from.

The final initial variable that is needed for each particle is velocity which is generated from the flow rate equation above. In this case the velocity of flow through a column's face is found to be ζ/A where A is the face area of the water surface within a column. Flows within the column structure may mean the velocity should not be perpendicular to the front of the face; to account for this the average flow from surrounding columns is used to give a direction to the scalar velocity calculated above. In many cases the velocity imparted to the particle should not only be horizontal but also include an initial vertical velocity. 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. While only an approximation this approach does manage to provide a more believable representation.

One of the methods that were initially considered to help create the illusion of water pooling and incompressibility was the use of cohesion. Intermolecular bonds are simulated by creating small forces between water particles which attract and repel neighbors in the effort of keeping an optimal distance apart. However, we found that using cohesion within the particle system is inefficient and inaccurate. Furthermore other authors suggest that cohesion is unnecessary due to the water's low viscosity so long as the movement is turbulent, such as is the case with spray [Sims 1990; Stein and Max 1998]. Consequently this extension was excluded when the particle and column systems were combined and the only active force during a time step is gravity.

It remains to deal with the collision detection for particles. The collision detection with the water volume boundary is explained in the next section. Collisions of particles with columns are detected by computing the column the particle is above (or within) and by comparing the particle's y-coordinate value to the column height in order to determine whether it should be absorbed into the column. However, simply increasing the column's volume and destroying the particle was found not to be accurate enough as this can force the column's height up, often absorbing more particles in the process. While this is not a problem for small scale splash effects considerable problems occur when modeling, for example, waterfalls where there are many particles hitting at any one time. After trying to spread the volume of a colliding particle over several columns it was found that by instead modeling the force of impact and subsequent pressure increase in the column not only was the problem reduced but more realistic effects were generated. The equations used are the same as for external objects colliding with the water presented in Mould and Yang [Mould and Yang 1997]. The force on the object consists of two terms:

$$F = -v\mu - V\rho g$$

where v is the velocity of the object, μ is the viscosity, V the volume of water displaced, ρ the density of the fluid and g the acceleration due to gravity. The first term describes the force of the fluid on the particle and the second is the force due to buoyancy. Because the forces on the fluid must be equal but oppositely orientated to that on the water droplet, this formula can then be used to determine the force on the column. Using the formulas:

$$P = m/A \quad \text{and} \quad m = \frac{F}{a}$$

we can calculate the resultant pressure of this force on the column, stopping it from rising unrealistically.

4 Implementation

The previous section presented the underlying physics of our modeling approach. This section explains several implementation details.

4.1 Particle System

Particles have a position, velocity, and mass but no volume. They can be affected by forces, either from outside the system or from other particles and can represent anything from rigid structures to fluids (as described earlier).

Our particle system is based on the work by Witkin [Witkin 1994] and was written as general as possible to allow for later extensions. All forces and environmental constraints are represented by interfaces that can be implemented as required. The system itself implements the interface required by the solver and as such represents itself as a point moving through $6n$ dimensional space where n is the number of particles and the position and velocity of each particles is represented by a 6-dimensional vector. The derivative of each particle's state vector $[x_1, x_2, x_3, v_1, v_2, v_3]$ is $[v_1, v_2, v_3, f_1/m, f_2/m, f_3/m]$ where f is the force acting on the particle. By representing the system as a point in $6n$-dimensional space, all forces between particles can be applied simultaneously so that the system stays consistent for each time step.

Solving for a particular time step begins with the system calculating the initial forces acting on the particles. The particle system can then pass itself to the solver method of choice. Because the solver views f(x,t) as a black box (i.e., it doesn't know how the function is evaluated) it calls back for solutions at intermediately positions. At these points the system recalculates the forces and responds appropriately. After the new positions and velocities are determined collision detection with the environment is performed. Panels defined in the environment file are used as boundary conditions to stop particles escaping. While any parallelogram is allowed as a panel it is important to note that the collision detection used here is only valid for rectangles. To find if a particle has passed a panel our algorithm checks first using the following formula which side of the

panel the particle is on:

$$(\mathbf{p}-\mathbf{v}_1)\cdot n \leq 0$$

where P is the particles position, $\mathbf{v}_1$ is one of the vertices of the panel and n is the unit normal of the plane on which the panel rests. A further check is necessary to see if the particle also lies within the rectangle described by the panel. If the four conditions below hold true then the point is within the parallelogram:

$$\begin{aligned}(\mathbf{p}-\mathbf{v}_2)\cdot(\mathbf{v}_2-\mathbf{v}_1) &< 0\\ (\mathbf{p}-\mathbf{v}_2)\cdot(\mathbf{v}_2-\mathbf{v}_3) &< 0\\ (\mathbf{p}-\mathbf{v}_4)\cdot(\mathbf{v}_4-\mathbf{v}_1) &< 0\\ (\mathbf{p}-\mathbf{v}_4)\cdot(\mathbf{v}_4-\mathbf{v}_3) &< 0\end{aligned}$$

where $\mathbf{p}$ is the particle's position and $\mathbf{v}_1,\ldots,\mathbf{v}_4$ are the four vertices of the rectangle in anticlockwise order. This is a simplification of the equation to check which side of a line a point is. As the sign is the only thing we are interested in it is more efficient to disregard excess calculation. A final check is made to ensure the particle is not "legally" on the other side of the panel. By tracing the particles path back one time step and checking that the old position was " correct" it can be assumed that the particle has indeed hit the panel.

As described in the previous section particles also have to be tested as to whether they have hit the main volume of water. Because columns are stored in a 2D array in their physical order the index (i,j) of the column where the particle k is above is computed by

$$(i,j) = \big((x_k - x_{min})N,\ (z_k - z_{min})N\big)$$

where (x_k, y_k, z_k) is the particle's position, x_{min} and z_{min} are the system's minimum x and z coordinates, respectively, and N is the number of columns per unit length.

If the particle is outside the bounds of the column array or its vertical position (y-coordinate value) is below all columns in the system the particle is destroyed. Otherwise it is absorbed into the volume model as described previously. Finally each particle is checked to ensure it hasn't exceeded its lifetime after which the particle system clock is increased by the time step given and the system is again deemed stable. Initially, when the particle system was being used exclusively, it was considered appropriate to try to model cohesion between the particles to mimic molecular interaction. The first attempt was to apply a simple spring with a limited area of effect. This spring forced particles towards an equilibrium situation. This then evolved through the application of the work done by Miller and Pearce [Miller and Pearce 1989] to that shown in figure 2. These forces were not found to be appropriate however as they made the fluid appear too viscous to be believable for water. They were also very slow.

4.2 Column System

As discussed above the volume model follows that used by Mould and Yang [Mould and Yang 1997] closely. Before the system can be used columns must be generated from the environment specified and pipes must be created between all adjacent.

Using the same environment file format as that originally used for the particle system, each shape is cycled through checking if columns should be created. Spheres, used to represent rocks and obstacles are considered first. For each sphere in the environment a cross-section along the xz-plane is used. Taking the lower left corner of a bounding rectangle each possible column position is checked to determine if it is within the cross-section. For this an approximation is used where each corner is checked to see if it is inside, if two or more corners are inside then a column is created in that position. The base height of each column created is found by determining the height of the center as it would be projected onto the sphere.

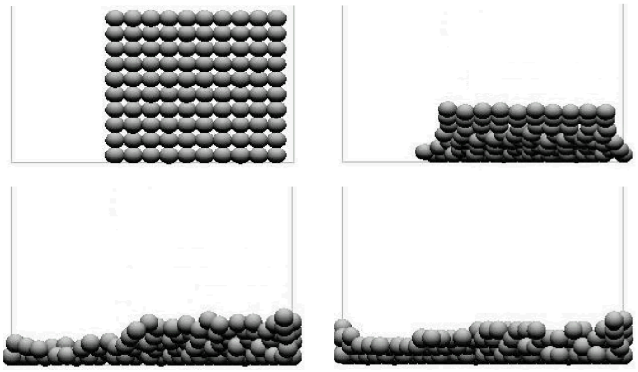

Figure 2: Particles (spray) with cohesion (left to right and top to bottom).

Each panel is also cycled through to create columns although, unlike spheres, each panel can serve one of three different purposes. If the area, on the xz-plane, is zero and the panel is not a source panel it is treated as a boundary panel for particle collision detection and is stored for use as such. Source panels also have no area but columns are still created along the line described by the first and second vertices. These columns have an initial height equal to the difference between the first and third vertices' y values which is kept regardless of outflows. However, most panels act in much the same way as all spheres by providing the ground environment for the column system. In order to create columns the algorithm checks all possible column positions within the bounding box of the projection of the panel onto the xz-plane fits. To do this opposite corners of the column position are tested as to whether they are left of the lines $\overline{\mathbf{v}_1\mathbf{v}_2}$, $\overline{\mathbf{v}_2\mathbf{v}_3}$, $\overline{\mathbf{v}_3\mathbf{v}_4}$, and $\overline{\mathbf{v}_4\mathbf{v}_1}$. Each time a score is incremented for each corner to the left of each line; if both corners are outside a line then no column is created otherwise if the "score" is above six (the opposite corners are outside of at most two lines) then a column is created as appropriate. This method is only an approximation but serves well in practice.

After each shape has been cycled through we have an unordered array of columns. By sorting them into a 2D array so they are placed according to their physical position a much better representation of the environment can be made, and more efficient algorithms can be used later. If there are two columns for any position then either the higher or source column is taken.

Because the columns are stored in order the creation of pipes is rudimentary. Each column is cycled through creating pipes in the directions shown in figure 1 (b). As described in the model section all possible pipes are created and then checked for validity later instead of dynamically creating pipes in each time step. To increment the system each pipe is cycled through with the intention of calculating flow. If a pipe's cross-section is less than 0 (i.e., the two cells don't overlap) nothing occurs, for those that do the flow and volume to be moved are calculated and the cells are updated accordingly. The user can choose the computational time step whereas the display time step is set to 0.005.

5 Rendering

5.1 Rendering the Water Surface

Since the column heights in our model are represented by a 2D array of points a smooth representation of the water surface is ob-

tained by using these points as the control points of a uniform B-Spline surface. Figure 3 shows an example. Note that when using a B-Spline interpolation the surface does not go through each control point but is instead "pulled" toward them. However, for low order B-Splines this approximation error is small. Since B-Splines have numerous advantages over alternative interpolation methods, such as fast rendering, local control and a high degree of continuity [Cohen et al. 2001] the representation seems to be the most appropriate for our task.

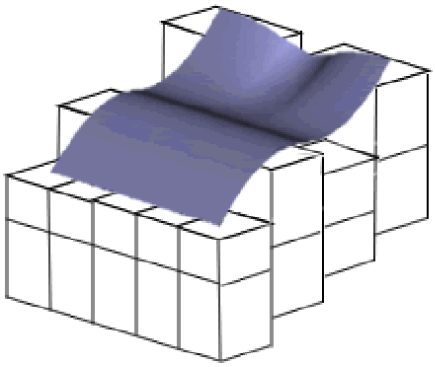

Figure 3: Uniform B-Spline surface fitted to the water columns.

The B-Spline surface is rendered by tessellating it into polygons. A fast display is achieved by using vertex arrays and display lists. The tessellation of the B-Spline surface requires the computation of surface points. For a regular sample grid the height $h(u,v)$ of each surface point can be computed by using a B-Spline reconstruction filter

$$h(u,v) = \sum_{i=\lceil u-m/2 \rceil}^{\lfloor u+m/2 \rfloor} \sum_{j=\lceil v-m/2 \rceil}^{\lfloor v+m/2 \rfloor} B^m(u-i)B^m(v-j)h_{ij}$$

where m is the order of the B-Spline reconstruction filter and h_{ij} is the column height at the grid point (i, j). In our application we use a quadratic B-Spline filter which is defined by

$$B^3(x) = \begin{cases} \frac{(2x+3)^2}{8} & -1.5 \le x < -0.5 \\ \frac{3}{4} - x^2 & -0.5 \le x < 0.5 \\ \frac{(2x-3)^2}{8} & 0.5 \le x < 1.5 \\ 0 & \text{otherwise} \end{cases}$$

Figure 4 illustrates a linear, quadratic and a cubic B-Spline filter.

Experimenting with the different order of the B-Spline filter suggested that 3^{rd}-order (quadratic) B-Splines are the most suitable ones for our applications since they are sufficiently smooth while still closely approximating the column positions. Figure 5 shows a comparison of different order B-Spline surfaces. It can be seen that linear B-Splines produce an in unrealistically rough surface whereas 3^{rd} and 4^{th}-order B-Splines result in surface details being lost. Higher order B-Splines also require more computation time since the corresponding reconstruction filter has a larger support.

5.2 Improvements

In order to simulate water flowing down a dry river zero column height values must be possible. We avoid erroneous zero height water surfaces by moving control points slightly below the base surface (and therefore out of view) if a column's height falls below a threshold.

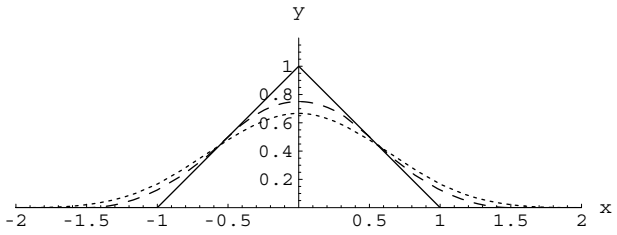

Figure 4: Graphs of the linear B-Spline filter (solid line), the quadratic B-Spline filter (dashed line) and the cubic B-Spline filter (dotted line).

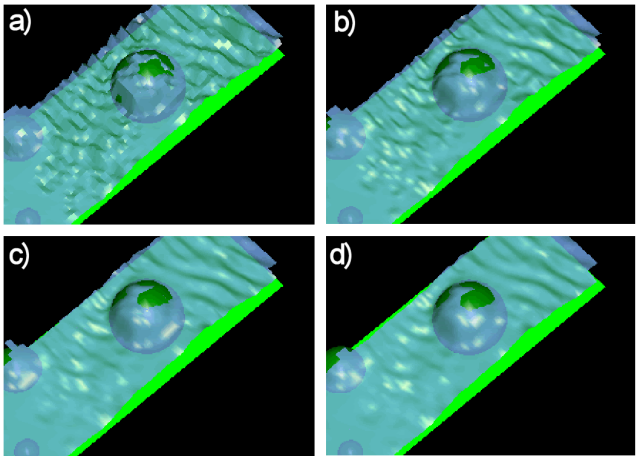

Figure 5: The water surface approximated with B-Spline surfaces of degrees one (a), two (b), three (c) and four (d).

One interesting benefit of using B-Spline surfaces is that nice effects can be achieved by offsetting control points slightly depending on the position of each column. This enables the simulation of adhesion along boundaries and possibly viscous shear, two things currently missing from the underlying model. Figure 6 shows an example.

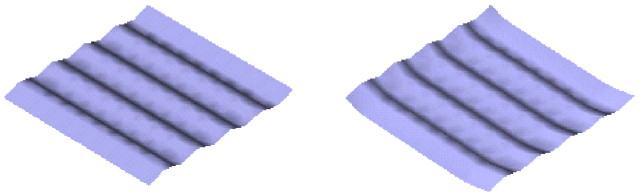

Figure 6: Original water surface (a) and the same surface with control points along the edges moved against the flow (b).

5.3 Ray Tracing for Increased Realism

The methods presented so far allow fast modeling and rendering of turbulent water. However, some application require more realistic images and we therefore also implemented a ray tracer for our model. Rather than ray tracing the tessellated B-Spline surface the B-Spline surface is ray traced directly using a technique suggested by Martin et al. [Martin et al. 2000].

The method decomposes a ray into two planes $\mathbf{P}_1 = (\mathbf{N}_1, d_1)$ and $\mathbf{P}_2 = (\mathbf{N}_2, d_2)$, the intersection of which is the original ray. The intersection points with the B-Spline surface $\mathbf{S}(u,v)$ are the roots of the function

$$\mathbf{F}(u,v) = \begin{pmatrix} \mathbf{N}_1 \cdot \mathbf{S}(u,v) + d_1 \\ \mathbf{N}_2 \cdot \mathbf{S}(u,v) + d_2 \end{pmatrix}$$

A Newton solver [Press et al. 1992] is then used to iterate an initial estimate for the intersection point with the B-Spline surface until the solution is within an acceptable error. If the surface tangent at an estimate is parallel to the ray no intersection between the tangent and the ray planes is found and the solver fails. This occurs if the Jacobian matrix $\mathbf{J} = (\frac{\partial \mathbf{F}}{\partial u}, \frac{\partial \mathbf{F}}{\partial v})$ is singular and in this case the estimate is moved randomly a short distance away.

Our implementation, while heavily reliant on [Martin et al. 2000], differs from it due to the unique characteristics of our model. First note that we can simplify the expressions for $\mathbf{F}$ and its partial derivatives because in our case the B-Spline surface is a height field. Also before employing the Newton solver we use an axis aligned bounding box to check if the ray comes anywhere near the water surface. Since the columns are axis aligned this check also returns the first column to be tested for intersection with the ray. The scene is the traversed in ray direction column by column until the first ray intersection with the water surface is found (see figure 7).

Since the B-Spline surface must be within the convex hull of its control points (i.e., the column heights) the ray surface intersection can be avoided in many cases. Otherwise the solver performs up to five iterations and if no solution is found the next column is checked.

5.3.1 Schlick's Approximation

Water is a *dielectric* (not to be confused with conductive properties) which means that the intensity of the refracted versus the reflected ray varies according to the angle of incidence. In general as the angle of incidence (θ) approaches zero the intensity of the refracted ray equals the intensity of the original ray and the intensity of the reflected ray goes to zero. The Fresnel equations are commonly

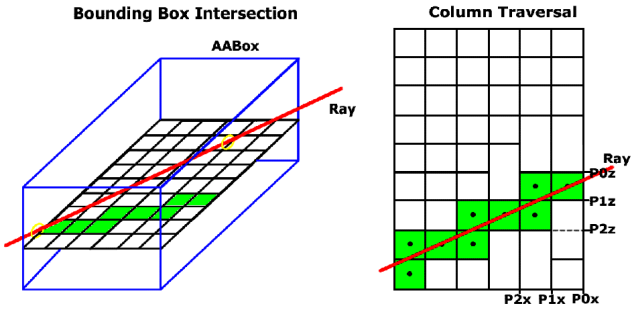

Figure 7: Traversal of water columns for ray tracing.

used to describe this relationship but are difficult to compute. As such, the standard equations used in ray tracing are those proposed by Schlick in [Schlick 1994] and are referred to as *Schlick's approximation*. The approximation does not provide exact answers and is only valid for unpolarized light but is suitable for our application [Shirley and Morley 2003]. The equations describing the relationship for reflectance (R(θ)) and transmittance (T(θ))are

$$R(\Theta) = R_0 + (1 - R_0)(1 - \cos\theta)^5$$

and from the conversation of energy

$$T(\theta) = 1 - R(\theta)$$

where R_0 is the base reflectance set by the user and $\cos\theta$ is computed from the dot product of surface normal and ray direction making the formula computationally inexpensive. It should be noted that this relationship is only used for transparent surfaces where refraction is expected to occur in the model and its effect is discounted for the ground panels and spheres etc. An example is shown in figure 8.

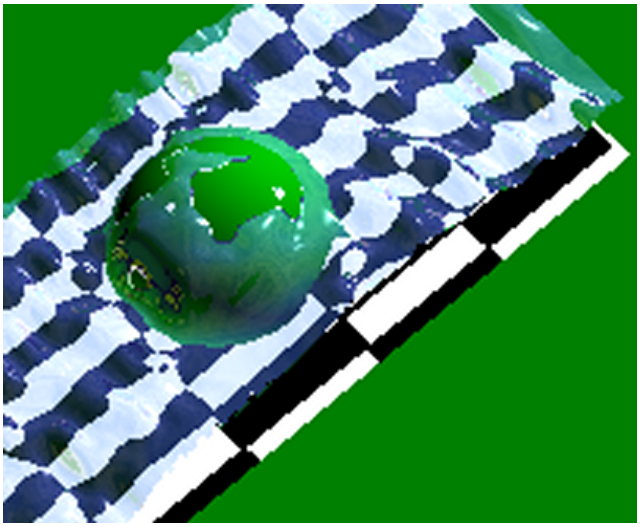

Figure 8: Refraction and reflection illustrated by using a ground plane with a checkerboard pattern.

5.4 Rendering of Particles

The particle system in our model represents water which breaks away from the water surface. Examples are small whitecaps on the

tops of waves, spray from water impacting on rocks, and laminar-like flows over waterfalls. So far we have implemented two basic methods shown in figure 9.

The representation in (a) uses small spheres with the radius (and therefore the size) determined by the volume of water that the particle represents. Some effort was made to differentiate between types of spray by using the velocity and "timeAlive" attributes of each particle to alter the color and opacity of the representing sphere. However the achieved effects were disappointing.

When using large numbers of particles ray tracing of the scene becomes especially slow and we use a space subdivision scene (BSP tree) to increase rendering speed.

Part (b) of figure 9 shows billboarded particles created by mapping a texture on a plane parallel to the view plane. Currently we draw a particle as a circle with the radius again dependent on the volume of water represented by it. In future we hope to achieve more realistic effects by evolving the texture maps during the animation (e.g., the texture stretches in the flow direction).

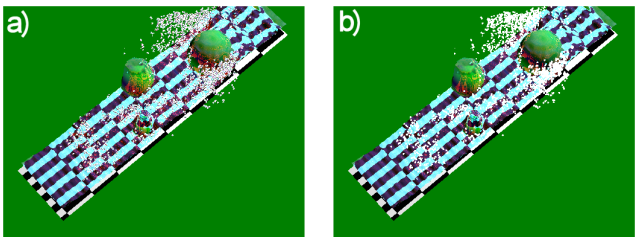

Figure 9: Particles rendered with spheres (a) and billboards (b).

6 Results

Using hydrostatics and drawing on information from fluid dynamics the model described in this paper is capable of creating realistic effects for the simulation of rivers and waterfalls. The speed of the model depends largely on the number of columns needed to cover the area and the number of particles in the scene. For example the simple river scene shown in figure 9 runs on a 1.4GHz PC without optimizations at 1 frame every 15 seconds with 13000 columns, each with 4 cells and just over 3700 particles. Most of this time is required for computing the next time step whereas the rendering alone takes only about 1.2 seconds.

The main drawback of the current model is the lack of viscous shear forces which is needed for creating eddies and visual effects such as shock waves around rocks and near the edges of the river. Also we haven't yet incorporated spray from water particles hitting columns too hard. In reality this often causes a semi permanent area of white water and mist around the base of a waterfall.

7 Conclusion

Using hydrostatics and fluid dynamics our model is capable of creating recognizable effects for the simulation of rivers and waterfalls. The speed of the model depends largely on the number of columns needed to cover the area and the number of particles in the scene. We are currently implementing more advanced implicit ODE solvers and current research indicates that this might lead to an order of magnitude improvement in speed which gets close to achieving interactive speeds.

For real time applications a polygon renderer can be used to display the tessellated B-Spline surface. Texture splats seem to be the most promising approach for representing spray and foam. We are currently investigating the use of illuminated streamlines for rendering other particle effects such locally laminar flows which occur, for example, in some types of water falls.

A photo realistic representation can be achieved by ray tracing the scene. Current results indicate that this takes at least two order of magnitudes more time than the polygon rendering approach and it should only be used for the final production step of non-real time applications.

References

BADGER, W. L., AND BANCHERO, J. T. 1955. *Introduction to Chemical Engineering*. McGraw-Hill Book Co.

CHEN, J. X., AND LOBO, N. D. V. 1995. Toward interactive-rate simulation of fluids with moving obstacles using Navier-Stokes equations. *Journal of Graphical Models and Image Processing 57*, 2 (Mar.), 107–116.

COHEN, E., RIESENFELD, R. F., AND ELBE, G., Eds. 2001. *Geometric Modelling with Splines: An Introduction*. A K Peters, Ltd., June.

ENRIGHT, D., MARSCHNER, S., AND FEDKIW, R. 2002. Animation and rendering of complex water surfaces. In *Proceedings of ACM SIGGRAPH 2002*, Computer Graphics Proceedings, Annual Conference Series, 736–744. `http://graphics.stanford.edu/papers/water-sg02/`.

FOSTER, N., AND FEDKIW, R. 2001. Practical animation of liquids. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, 23–30.

KASS, M., AND MILLER, G. 1990. Rapid, stable fluid dynamics for computer graphics. 49–57.

MARTIN, W., COHEN, E., FISH, R., AND SHIRLEY, P. 2000. Practical ray tracing of trimmed nurbs surfaces. *Journal of Graphics Tools 5*, 1, 27–52.

MILLER, G., AND PEARCE, A. 1989. Globular dynamics: A connected particle system for animating viscous fluids. *Computers and Graphics 13*, 3, 305–309.

MOULD, D., AND YANG, Y.-H. 1997. Modeling water for computer graphics. *Computers and Graphics 21*, 6, 801 – 814.

NEYRET, F., AND PRAIZELIN, N. 2001. Phenomenological simulation of brooks. In *Computer Animation and Simulation '01*, Springer Computer Science, 53–64. Proceedings of the Eurographics Workshop in Manchester, UK, September 2–3, 2001.

O'BRIEN, J. F., AND HODGINS, J. K. 1995. Dynamic simulation of splashing fluids. In *Proceedings of Computer Animation '95*, IEEE Computer Society, 198–204. `http://www.cc.gatech.edu/gvu/animation/papers/water.pdf`.

PRESS, W. H., VETTERLING, W. T., TEUKOLSKY, S. A., AND FLANNERY, B. P. 1992. *Numerical Recipes in C - The Art of Scientific Computing*, 2nd ed. Cambridge University Press. URL: `http://www.library.cornell.edu/nr/bookcpdf.html`.

REEVES, W. T. 1983. Particle systems – a technique for modeling a class of fuzzy objects. *ACM Transactions on Graphics 2*, 2 (Apr.), 91–108.

SCHACHTER, B. J. 1980. Long crested wave models. *Journal of Computer Graphics and Image Processing 12*, 2 (Feb.), 187–201.

SCHLICK, C. 1994. An inexpensive BRDF model for physically-based rendering. *Computer Graphics Forum - Proceedings of Eurographics 1994 1*, 3, 233–246.

SHIRLEY, P., AND MORLEY, R. K. 2003. *Realistic Ray Tracing*, 2^{nd} ed. A K Peters, Ltd.

SIMS, K. 1990. Particle animation and rendering using data parallel computation. *Computer Graphics 24*, 4 (Aug.), 405 – 413.

STEIN, C. M., AND MAX, N. L. 1998. A particle-based model for water simulation. Technical report UCRL-JC-129378, Lawrence Livermore National Laboratory, Jan. URL: `http://www.llnl.gov/tid/lof/documents/pdf/233792.pdf`.

THORNTON, E. B., AND GUZA, R. T. 1982. Energy saturation and phase speeds measured on a natural beach. *Journal of Geophysical Research 87*, C12, 9499 – 9508.

WITKIN, A. 1994. Particle system dynamics. In *SIGGRAPH '94, course notes #32 - An Introduction to Physically Based Modeling*. July.