Research on the Rhine: Reflections on Water Simulation

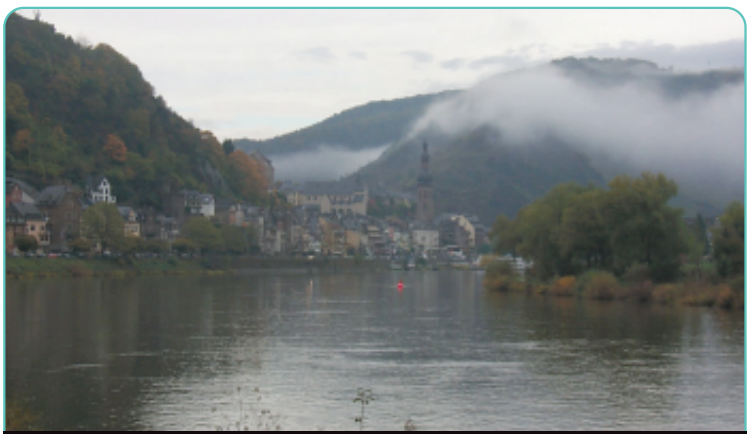
Last month I left off talking about what makes water look realistic in a simulation. I used image processing to create an effect that behaved something like the way water behaves. However, the technique wasn't based on any physical foundation. Now we need to consider some physical properties of liquid.

A topic as complex as the computer simulation of the behavior of liquids requires research. In fact, I have spent the last week traveling up and down the Rhine valley in western Germany observing one of the great rivers of the world. All right, so I mostly sat in a *Weinstube* watching the barge traffic travel up and down the Rhine. I just ordered another round of wonderful *Spätlese Rieslings* (*trocken* for me, *lieblich* for my wife) while we discussed the boat maneuvering between shallow water reefs in the river.

Germany is a very interesting place to research fluid dynamics. As a couple of native Californians, several physical realities were very clear to us. The only liquid readily available and affordable was wine (or beer in the rest of the country) **{confusing...the only avail. liquid was wine in the Rhine valley but there's beer in the rest of the country? plz. elaborate or clarify}** and the restaurants provided an amazing demonstration of the interaction of turbulent hot gases in a dynamic environment. The level of cigarette smoke allows you to observe completely the natural eddies and turbulent rotation that occurs as people travel through the field. As people who have been subject to California's rather draconian laws regarding the free release of turbulent hot gases in confined spaces, it was quite an impressive sight.

Watching the barges travel up and down the river, I was impressed by the size of the wakes left by these enormous vessels. The wakes interacted with the standing waves in the river, reflected off the banks, and generally interfered with each other creating complex patterns in the water. Where the river narrowed then widened again, large eddies formed along the banks slowly swirling around and around.

The connecting Mosel river winds its way up the canyons toward Belgium. To control the level of water along the river, a series of dams have been constructed, which the barges navigate through a series of locks. The barges enter the lock which is then sealed. The water is raised or lowered to match the level on the other side of the dam, and then the vessel continues on its way. It's a wonder of fluid dynamics to behold. These massive barges brimming with heavy cargo are lifted by simply pumping more water into the bathtub.



Kicking back in one of the Rhine valley's Weinstuben *offered the author a chance to contemplate realistic simulations of water behavior, among other things.*

μ	Viscosity of the fluid
ρ	Density of the fluid
∇	Gradient operation $= (d/dx)i + (d/dy)j + (d/dz)k$
g	Gravity vector
p	Pressure of a point in the fluid
v	Velocity vector
s	Height of water surface
b	Height of water bottom
d	Depth of the water

A summary of the notations used in this article.

Wenn er nicht die Stufe der Flüssigkeit fließend aus seinem Weinglas heraus erwägt, kann Jeff bei Darwin 3D erreicht werden. Überprüfen Sie, daß er wirklich zurück zu Arbeit erhalten hat, indem er ihm an jeffl@darwin3d.com schrieb.

This elegant method for transporting cargo has been in use in western Europe since the thirteenth century.

Where Is This Leading?

These things that I observed along the waterways of Europe are exactly the kind of effects I would like to build into my fluid simulations. The image-based method I developed last month didn't simulate the physical properties of water. There were ripples and they interacted with each other, but modeling something like an eddy was way beyond the capability of that simple technique.

What goes into making an eddy, anyway? First I need to discuss some physical properties of water. When a river is moving, the individual water particles are constantly interacting with each other. The particles rub against each other, against the sides of the river, and against any rocks or other obstacles in the flow. These interactions are a form of friction between the water particles. The physical property of the fluid that regulates the amount of friction interaction is called viscosity. You probably think of motor oil when you hear that term. However, viscosity is a way of describing the degree to which the particles will interact similar to the coefficient of friction in the Coulomb dry friction model **{Should we insert ref to tribology column?}**.

To better visualize this interaction, imagine that a river is a series of water layers like stacked blocks as you see in Figure 1a. When the flow is unobstructed, the layers all move together in the direction of the flow. However, when the flow of the bottom layer is obstructed, the friction force is transferred between the layers, slowing them and resulting in the situation you see in Figure 1b.

When a flow behaves in this layered manner, the flow is said to be laminar. Another form of flow is called turbulent flow. In a turbulent flow, particles belonging to different layers become mixed due to the internal friction of the flow. For my simulation, I will only be concerned with laminar flows.

Now if the river is flowing at relatively slow velocity, when it encounters a narrow section the viscous forces are transferred through all the layers of water, slowing the river down. You can see this in Figure 2a. However, if the inertial velocity of the water is strong enough to overcome the viscous forces acting between the layers, the flow separates from the shore. This creates an eddy or vortex rotating in the direction opposite the flow along the shore. In a river, this is where the water pools and debris and stagnant water accumulate. You can see this behavior in Figure 2b.

This form of physical realism creates a much more interesting game simulation. Clearly, I would like to calculate the effects of viscous forces in my virtual water. To do this I need to turn to computation fluid dynamics.

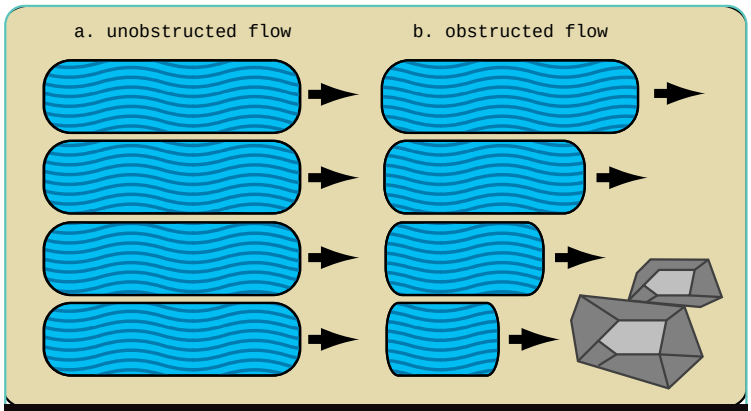


FIGURE 1. *Water layers, shown unobstructed (A) and obstructed (B).*

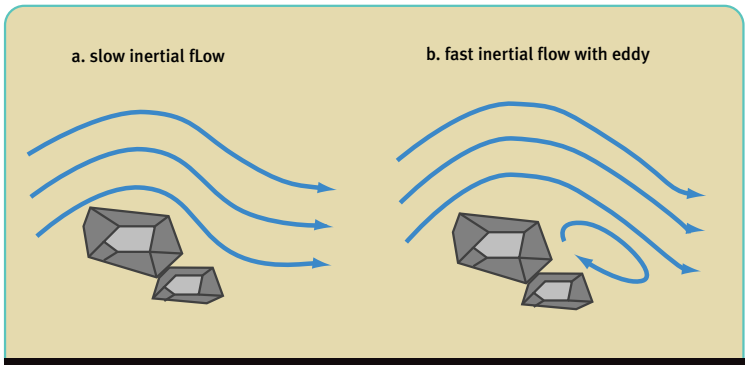


FIGURE 2. *A slow internal flow (A) and a fast internal flow (B), creating an eddy.*

CFD and Gaming

Engineers have been studying fluid flows for a long time now. Computation fluid dynamics, or CFD, has been a very important field with a great variety of applications. Aircraft and automobile manufacturers study the flow of air across the surfaces of planes and cars. Mechanical engineers study the flow of liquids through pipes and structures like dams. Even rocket scientists use fluid dynamics to understand the flow of the jet engines.

Fluid particles behave according to the laws of physics. In general, particles behave according to Newton's second law. The second law states that the sum of forces acting on a particle is equal to the rate of change of the linear momentum of the particle. If the mass is constant, the sum of the forces is equal to the product of the particle's mass and acceleration. In mathematical terms, this is the famous $F = ma$.

In the nineteenth century, physicists Navier and Stokes applied Newton's second law to the field of fluid dynamics. The behavior of a fluid particle is governed by a series of equations called, not surprisingly, the Navier-

Stokes equations. The first equation describes the force acting on an infinitesimal fluid particle.

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

This equation states that the acceleration of the fluid particle is a function of the pressure, velocity, and force of gravity acting on it. The formula is valid for fluids that exhibit a linear relationship between the pressure components and the velocity gradients. These fluids are called Newtonian fluids and include most common fluids such as water, oil, and air. **{Air a fluid? I learn something new every month...}**

A second part of the Navier-Stokes equations enforces the fact that Newtonian fluids are incompressible. That is, the mass of the fluid must be conserved.

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0$$

These equations describe the complete behavior of Newtonian fluids. However, it is fairly complex. In the field of CFD it's necessary to have highly accurate simulations regardless of computational expense. As we know, in the field of real-time computer graphics, we do not necessarily share these priorities. As game developers, we are perfectly willing to sacrifice correctness in exchange for interactive rates. Our priorities are to create a realistic-looking animation using the fastest calculations possible.

The Navier-Stokes equations describe the motion of the entire fluid field in three dimensions. However, for most applications, I am not really concerned with the interactions within the fluid. The key interaction for visual realism is the surface of the fluid. By reducing the simulation to a 2D problem, I can greatly reduce the calculations required.

A Watery Landscape

Michael Kass and Gavin Miller realized this and tried to simplify the Navier-Stokes equations. Physics have **{'Physics has' or '{Physicists have?'}** used a simplification of the above equations to predict the motion of shallow water. The key was to make several assumptions.

The water surface should be thought of as a height field. This restricts the possible range of effects, as you cannot have splashing or breaking waves. This is similar to the use of height fields for terrain landscapes. In a height field terrain, it is not possible to have overhangs or caves without resorting to multiple layers or other tricks.

The second assumption is that the horizontal velocity of the water is constant throughout the column. This would not accurately simulate the ground friction I showed in Figure 2. However, this is not really important to the surface appearance of the water.

The third simplification is that the vertical velocity of the water particles is ignored. This will lead to problems only if the change of height in adjacent columns occurs too dramatically. However, in practice, this is not a big issue.

Figure 3 shows a one-dimensional water height field. The height of the surface of the water is $s(x)$, and $b(x)$ is the height of the water bottom. I will set $d(x) = s(x) - b(x)$ to be the depth of the water at location x. The velocity of the column is given by $v(x)$.

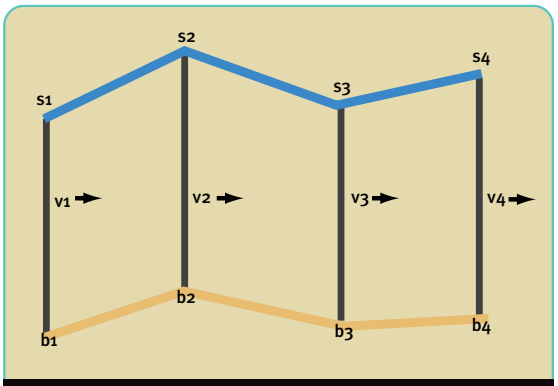


FIGURE 3. *A one-dimensional water height field.*

Given all these assumptions, the Navier-Stokes equation for shallow water fluid flow becomes

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + g\frac{\partial s}{\partial x} = 0$$

$$\frac{\partial d}{\partial t} + \frac{\partial}{\partial x}(ud) = 0$$

The first equation is Newton's second law and the second equation is the conservation of mass. Kass and Miller simplify this further by eliminating the second term of the first equation and change the second equation to vary with the surface height. This assumption causes the speed of propagation to be constant throughout the field, which should not be a problem if the fluid velocity it relatively small.

$$\frac{\partial u}{\partial t} + g\frac{\partial s}{\partial x} = 0$$

$$\frac{\partial s}{\partial t} + \frac{\partial}{\partial x}(ud) = 0$$

These equations can be combined as shown:

$$\frac{\partial u}{\partial t} + g\frac{\partial s}{\partial x} = 0$$

$$\frac{\partial^2 u}{\partial t \partial x} + g\frac{\partial^2 s}{\partial x^2} = 0$$

$$\frac{\partial s}{\partial t} + \frac{\partial}{\partial x}(ud) = 0$$

$$\frac{\partial^2 s}{\partial t^2} + \frac{\partial^2}{\partial x \partial t}(ud) = 0$$

$$\frac{\partial^2 s}{\partial t^2} = gd\frac{\partial^2 s}{\partial x^2}(ud)$$

This last equation results in a partial differential equation. In order to be useful, I need to change that to a discrete form. Kass and Miller suggest the use of finite-difference techniques for a delta time h in the form:

$$\frac{\partial^2 s}{\partial t^2} = -g\left[\frac{d_{x-1} + d_x}{2(\Delta x)^2}\right](s_x - s_{x-1}) + g\left[\frac{d_x + d_{x+1}}{2(\Delta x)^2}\right](s_{x+1} - s_x)$$

This final formula describes the vertical acceleration of the water surface at position x. I can now simulate the surface of the water in 2D. The formula also largely conserves mass as the mass conservation formula was calculated into the final equation. However, there is one situation that is not handled — there is no restriction that the surface of the water must be as high as bottom surface, meaning $s < b$ is possible. This requires a bit of tweaking to fix. The solution is to conserve volume manually for any location where $s < b$ by checking the water depth in the previous frame in the location and its neighbors. If the volume differs from frame to frame, I can distribute the difference into the neighboring cells. This formula is easy to code up and stick into my existing simulation framework. This makes it easy to start playing around with variations on the formula.

I am really interested in making the simulation in 3D. Luckily it is easy to extend the algorithm. First, the height field is extended to a 2D array. This array is evaluated in the x and z directions using two passes with the finite-difference equation. The water height is determined in the y direction and, finally, the grid is sent as a series of triangle strips to the renderer.

Another Approach

One problem with the above approach is that the shallow water simplifications eliminate some of the features of the original Navier-Stokes equations. One particular change was the elimination of the viscosity factor from the equation. The assumption was that the entire fluid field was a constant viscosity. For some simulations, it may be desirable to have fluids with viscosity values that vary over time. Likewise, the assumption that the wave propagation speed is constant could be a problem for some simulations.

Jim Chen and Niels Lobo approached the problem by using the Navier-Stokes equations more directly. The solution they proposed was to consider the fluid field as a 2D grid as Kass and Miller did. However, in their approach the height of the cell under consideration is not determined directly. They start with the general Navier-Stokes equation.

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

This formula is used to generate the velocity vectors and pressure values at every point in the grid. To determine the height in the y direction of a particular point on the grid, the pressure at that point is considered. The height is then determined as some scaled value of this pressure reading.

This method effectively allows you to simulate the interaction of fluids with different viscosities. However, there is no current consideration for the shape of the water floor. Since I find this to be an important aspect for many game simulations, I will have to determine how that can be added into the equation without considering the complete 3D Navier-Stokes equations.

Obviously, completely voxelizing my simulation area and then calculating the complete equations at every point in the environment would be ideal. This would allow for very realistic fluid that could splash and break on itself, as well as swirl at every level throughout the volume. That is exactly what Nick Foster and Dimitri Metaxas have done. Their simulation allows for completely realistic simulation of fluid in 3D. However, the voxelized space that they simulate is fairly small and even that does not run anywhere near interactive rates. Jos Stam proposed another method for computing the fluid dynamics in a 3D field at Siggraph 99. But once again, the field size was much too small to be usable in any of the applications I have in mind.

So for now at least, I am left with height map techniques for interactive simulation. I will continue to research the topic and keep you updated. At least the computers are getting faster; that will certainly help. But for now, I'm going to get back to enjoying my wine. ■

FOR FURTHER INFO

- Griebel, Michael, Thomas Dornseifer, and Tilman Neunhoeffer. Numerical *Simulation in Fluid Dynamics: A Practical Introduction.* Philadelphia: SIAM, 1998.
- Kass, Michael, and Gavin Miller. "Rapid, Stable, Fluid Dynamics for Computer Graphics." *Siggraph 1990* Vol. 24, No. 4: pp. 49–55.
- Chen, Jim, and Niels da Vitoria Lobo. "Toward Interactive-Rate Simulation of Fluids with Moving Obstacles Using Navier-Stokes Equations," *Graphics Models and Image Processing* Vol. 57, No. 2 (March 1995): pp. 107–116.
- Foster, Nick, and Dimitri Metaxas. "Realistic Animation of Liquids," *Graphics Models and Image Processing* Vol. 58, No. 5 (1996): pp. 471–483.
- Stam, Jos. "Stable Fluids." *Proceedings of Siggraph 1999.* New York: ACM Siggraph, pp. 121–127.