

Real-Time Fluid Simulation using Discrete Sine/Cosine Transforms

Benjamin Long*
University of Bristol

Erik Reinhard†
University of Bristol

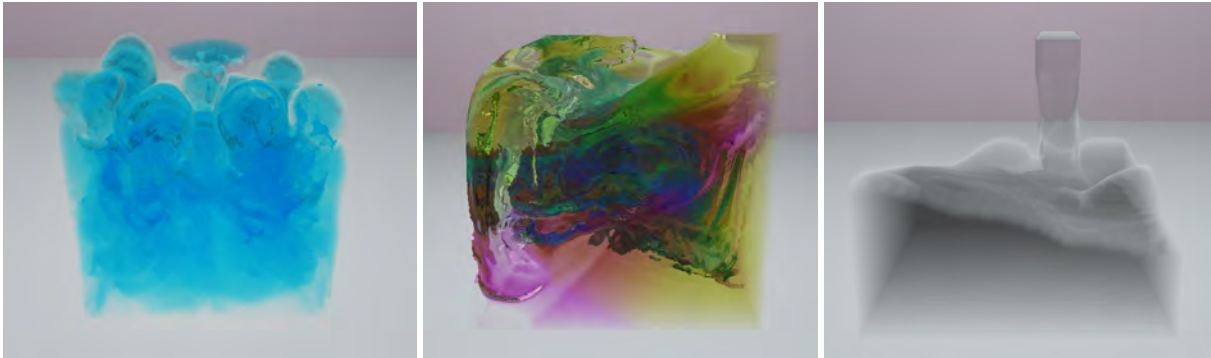


Figure 1: An example of our real-time fluid simulator, showing Rayleigh-Taylor instability simulation (left), a multi-liquid simulation (middle), and pouring water (right). See also Color Plate 1.

Abstract

Recent advances in fluid simulations have yielded exceptionally realistic imagery. However, most algorithms have computational requirements that are prohibitive for real-time simulations. Using Fourier based solutions mitigates this issue, although due to wrap-around, boundary conditions are not naturally available, leading to inconsistencies near the boundary. We show that slip boundary conditions can be imposed by solving the mass conservation step using cosine and sine transforms instead of the Fourier transform. Further, we show that measures against density dissipation can be computed using cosine transforms and we describe a new method to compute surface tension in the same domain. This combination of related algorithms leads to real-time simulations with boundary conditions.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically-based Modeling;

Keywords: Real-Time Fluid Simulation, DCT/DST Transforms, Real-Time Rendering

1 Introduction

The simulation of fluids is an active field of research, and has led to several important applications. For instance, the prediction of weather relies on such simulations [Denis et al. 2001]. The simulation of smoke [Foster and Metaxas 1997; Stam 1999; Fedkiw et al. 2001] in an environment can be important to simulate for the purpose of understanding, for instance, how a building can be most

efficiently evacuated in case of fire. Finally, rendering the results of such simulations is useful in the visual effects industry.

Many algorithms for fluid simulations have recently appeared in computer graphics, and include interesting effects such as multiple liquids [Losasso et al. 2006], coupling with solid objects [Géneveaux et al. 2003; Carlson et al. 2004; Guendelman et al. 2003; Batty et al. 2007] and the simulation of fire [Lamorlette and Foster 2002; Nguyen et al. 2002].

Liquids are volumetric phenomena, with each part of the volume interacting locally, producing streams and rotational effects such as vortices and eddies. This behavior can be captured with the well-known Navier-Stokes equations, introduced to computer graphics by Kayija and Herzen [1984], which describe how a velocity field changes over time as a result of applying pressure and forces. The volume is then discretized into voxels, and these equations are solved simultaneously for each voxel.

Thus, the Navier-Stokes equations show how the velocity \mathbf{u} of the fluid changes over small time steps t , as function of the pressure p and density ρ , the viscosity of the fluid μ and a general force \mathbf{f} which can model various desirable features, such as for instance gravity and surface tension. For each position in the fluid, the Navier-Stokes equations are given by:

$$\frac{\partial \mathbf{u}}{\partial t} = -\mathbf{u} \cdot \nabla \mathbf{u} - \nabla p + \mu \nabla^2 \mathbf{u} + \mathbf{f} \quad (1)$$

$$\nabla \cdot \mathbf{u} = 0 \quad (2)$$

The transport of fluid, termed advection, is modeled by (1), whereas the conservation of mass is modeled by (2). The latter ensures that if fluid dissipates from a region, it is replenished by an equal amount from neighboring regions.

For fluids that carry with them additional particles, for instance smoke particles in a smoke simulation, it may be necessary to compute a density distribution ρ_s . Such a scalar field is related to velocity only, and is given by:

$$\frac{\partial \rho_s}{\partial t} = -\mathbf{u} \cdot \nabla \rho_s \quad (3)$$

While the Navier-Stokes equations determine how fluid behaves over small time-steps, we also need to discretize the fluid into small

*e-mail: long@cs.bris.ac.uk

†e-mail: reinhard@cs.bris.ac.uk

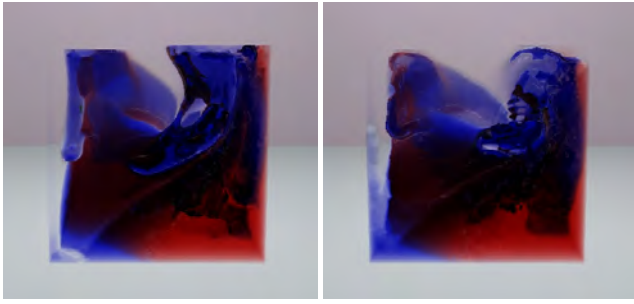


Figure 2: Advection computed with MacCormack (left) and semi-Lagrangian methods (right). See also Color Plate 2.

voxels, and solve these equations simultaneously for each voxel. For simplicity, often a rectilinear grid is used, although irregular and even dynamic meshes are also possible. This allows the mesh to adjust to the flow of fluid, for instance in the presence of solid obstacles [Feldman et al. 2005; Klingner et al. 2006].

To enable large time steps to be used for the purpose of speeding up the computation, the advection equation can be solved with semi-Lagrangian methods [Stam 1999; Anderson 1995; Selle et al. 2007] or Back and Forth Error Compensation and Correction (BFEC) [Dupont and Liu 2003; Kim et al. 2005]. As semi-Lagrangian approaches tend to lose small-scale detail, vorticity confinement methods can be applied to amplify remaining detail [Fedkiw et al. 2001]. Alternatively, particle methods can be incorporated into the solution to avoid loss of detail [Selle et al. 2005]. In our work, shown in Figure 2, we adopt the use of a straight semi-Lagrangian method and a hybrid (MacCormack) method [Stam 1999; Selle et al. 2007], as these are the fastest available solutions, and are amenable to real-time solutions. Vorticity confinement is integrated to provide control over small-scale detail.

The mass conservation step can be computed by finite differencing schemes coupled with a sparse linear system solver such as Gauss-Seidel, Jacobi, or SOR [Strang 1988]. The advantages of these approaches are relative ease of implementation, and good numerical stability. Unfortunately, they are only suitable for off-line simulations, due to their high computational complexity.

To achieve interactive simulation rates, currently only a few different approaches are possible. One approach is to simplify the problem so that only volumes near the surface boundary are simulated at high accuracy, with the remainder of the volume treated at a lower resolution [Irving et al. 2006]. It is also possible to apply dimensionality reduction to obtain approximate solutions throughout the volume, and thus obtain interactive simulation rates [Treuille et al. 2006].

Fourier-based solutions are fast enough for real-time simulations, and provide the same accuracy throughout the volume. They can be applied to the mass conservation step directly under the assumption that the volume infinitely repeats [Stam 2001], i.e., it cannot be naturally bounded. In Section 2 we review these techniques, as our approach relies on related transforms: we propose to replace the Fourier domain with discrete cosine and sine transforms, allowing us to enforce boundary conditions naturally.

Other fluid simulations and PDE solving algorithms have achieved this replacement [Bessonov et al. 1995], but use an indirect method involving building linear systems from finite differences and then preconditioning these systems to allow more powerful solvers such as the Fourier analysis cyclic reduction (FACR) algorithm [Hockney 1965] to be brought to bear on the original problem. The PDE

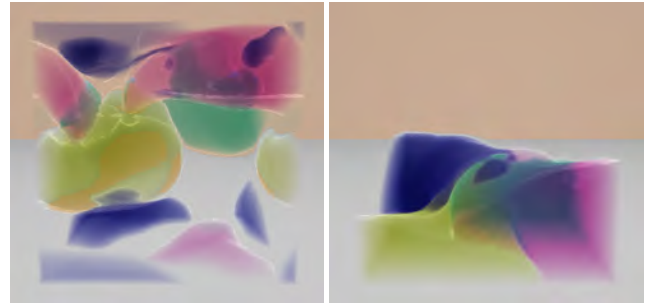


Figure 3: Images of a progressing simulation with gravity, using FFT (left) and alternatively DCT/DST (right). Notice that the fluid on the left is in free fall due to the wrap around effect of the FFT. See also Color Plate 4.

solver we use, the Fourier spectral fluid solver [Yokokawa et al. 2002], is direct but will only efficiently solve PDEs that involve simple operations in frequency space. As shown in Section 3, this solver can be used in a combined DCT/DST domain, which enables us to solve the fluid mass conservation equation using the boundary conditions that this domain provides (see Figure 3).

In comparison with spectral fluid simulations with different basis functions which also impose boundary conditions, such as the Chebyshev transform [Boyd 1999], this method has the advantage of being able to capitalize on a large body of research on fast DFT/DCT/DST transforms [Makhoul 1980; Martucci 1994; Strang 1999; Reeves and Kubik 2006], for instance in the form of the FFTW package [Frigo and Johnson 2005].

The second problem we address in this paper is that of density dissipation which occurs as a result of the discretized nature of the volume. Section 4 shows that a cosine transform can be used to smooth the scalar density field for the purpose of computing a gathering term which can then be added to the density distribution. This addition gently counters numerical dissipation effects over large spans of simulation steps.

The third problem addressed here is the computation of surface tension at real-time speeds. Normally, the surface curvature is computed at surface boundaries, and from this a force can be derived that is folded into the advection equation. Instead, in Section 5 we show that surface curvature can be estimated from the second derivative of the scalar density field, which in turn can be computed efficiently using the discrete cosine transform.

The fourth problem is that viscosity has been treated previously in this method as an approximating smoothing convolution which is applicable only to a single fluid. Section 6 addresses this by computing the contribution due to viscosity accurately in the DCT/DST frequency space and generalizing the resulting algorithm to multiple viscosities.

We show how our computations can be extended to multiple interacting liquids in Section 7. Details of our multi-threaded implementation are given in Section 8. To render our results in real-time we use a straightforward implementation of the ray equation, which is briefly discussed in Section 9. Finally, results are shown and discussed in Sections 10 and 11 respectively.

2 Fourier-Based Fluid Simulation

To achieve simulations that run at interactive rates, Fourier-space methods have been proposed [Stam 2001]. In particular, the mass conservation can be reformulated in Fourier space, noting that $\nabla \cdot \mathbf{u}$

in a grid of voxels is computed with central differencing. In one dimension this is given by:

$$\frac{\partial \mathbf{u}}{\partial x} = \sum_{n=1}^{\infty} (\mathbf{u}_{x+n} - \mathbf{u}_{x-n}) (-1)^n / n \quad (4)$$

Thus, we are computing the convolution of differences about \mathbf{u}_x with $(-1)^n/n$. In the Fourier domain, this is a computationally less expensive multiplication so that the mass conservation step is given by:

$$\frac{2\pi k}{K} \mathbf{U}_x(k, l, m) + \frac{2\pi l}{L} \mathbf{U}_y(k, l, m) + \frac{2\pi m}{M} \mathbf{U}_z(k, l, m) = 0 \quad (5)$$

or equivalently

$$\left(\frac{k}{K}, \frac{l}{L}, \frac{m}{M} \right) \cdot \mathbf{U}(k, l, m) = 0 \quad (6)$$

where we have divided out the factor of 2π . Note that \mathbf{U}_x , \mathbf{U}_y , and \mathbf{U}_z are separate 3D Fourier transforms, accounting for the fact that we wish to Fourier transform a vector field, rather than a scalar field.

A consequence of the Fourier space approach to solving the mass conservation step is that the solution is obtained throughout the volume simultaneously. This means that propagation effects due to unsteady pressure gradients ∇p cannot be accounted for. As a result, Fourier space methods are only applicable to fluids where either $\nabla p = 0$ or ∇p has a predictable distribution which can be simulated via \mathbf{f} . For the same reason, shockwaves cannot be blocked or redirected by boundaries, i.e. Fourier space methods exhibit a significant disadvantage in that boundary conditions cannot be imposed. A further disadvantage results from the fact that the domain in Fourier space replicates, resulting in an inaccurate smearing of detail across any boundaries attempted in the fluid.

In our work, the key innovation is that we provide a suitable alternative to the use of the Fourier transform so that natural boundary conditions are created, keeping the fluid simulation from exhibiting these undesirable behaviors. This is achieved without adding extra computational expense. We are therefore able to show real-time simulation of fluids which observe natural boundaries.

3 Mass Conservation

We will assume that the fluid will be represented by an axis-aligned rectilinear grid of voxels. The grid itself forms the boundary of the volume. It has inward pointing surface normals, which are assumed to remain stationary. This leads to the following slip boundary condition:

$$\mathbf{u} \cdot \mathbf{n}_i = 0 \quad (7)$$

where \mathbf{u} is the velocity vector of the fluid and \mathbf{n}_i is the normal vector of the i^{th} face. This equation prevents fluid flowing through a surface whilst enabling fluid flow along it. This is achieved by forcing the component of the velocity vector corresponding to the surface normal to zero near the boundary. As a result, the fluid approaching the boundary of the rectilinear volume will be deflected.

An alternative way to think about this is to assume that our domain is replicated, with neighboring instances being mirror images. Thus, if some amount of fluid would try to leak out of our domain, an equal and opposing force from a neighboring instance would prevent this, leading to a net velocity directed along the boundary.

Thus, if we were to enlarge our fluid by reflecting the volume in three orthogonal planes, we would effectively create boundary conditions. This would allow us to use the Fourier method to solve

the mass conservation step, but at the cost of a domain that is eight times as large, which requires a significantly longer computation time.

It can be shown that the discrete Fourier transform (DFT) of the enlarged volume can be replaced with a discrete sine transform (DST) of the smaller original volume [Makhoul 1980]. The DFT of a $2n$ -element reflected volume (also known as a $2n$ -point odd extension) will yield a $2n$ element complex field U , with only n of them carrying information. The DST of an n -element field will yield an n element real transform U^S , with all elements carrying information. The relationship between these two transforms is given by:

$$U^S(n-1) = iU(n) \exp\left(\frac{i\pi n}{N} \left(4N - \frac{1}{2}\right)\right) \quad (8)$$

where $n \in [1, N]$, which includes a half-sample offset [Strang 1999], a shift in indices, and a rotation to turn the complex values produced by the DFT U into real values associated with the DST U^S , in accordance with the conventions adopted for these types of transform. Similarly, we can construct a discrete cosine transform (DCT-II) U^C :

$$U^C(n) = U(n) \exp\left(\frac{i\pi n}{N} \left(4N - \frac{1}{2}\right)\right) \quad (9)$$

The original transformation to Fourier space involves a 3D DFT for each of the three components of the velocity vector \mathbf{u} , applied in three different dimensions. We now replace these transforms by a combination of DST and DCT transforms:

$$\mathbf{U}' = \left(\mathbf{U}_x^{\text{DCT}}, \mathbf{U}_y^{\text{DST}}, \mathbf{U}_z^{\text{DCT}} \right) \quad (10)$$

The superscripts indicate which transform is chosen for which dimension. As an example, the first component of the velocity field will be transformed with a DST in the x -dimension, and with DCT transforms in the y - and z -dimensions. Note that the computational complexity and memory requirements of this transform are equivalent to the original triplet of DFTs.

Since we are only interested in computing the three derivatives $\partial u_x / \partial x$, $\partial u_y / \partial y$ and $\partial u_z / \partial z$ we note that these derivatives are all in directions in which we have used the DST. For this reason we do not need to calculate derivatives in DCT space.

The partial derivative of u can be written as the inverse DST of U^S , with terms scaled by appropriately chosen constants. The mass conservation is then computed with:

$$\left(\frac{k+1}{K}, \frac{l+1}{L}, \frac{m+1}{M} \right) \cdot \mathbf{U}'(k, l, m) = 0 \quad (11)$$

With this approach, we have gained control over the fluid behavior near boundaries, yielding appropriate boundary conditions that were previously unavailable in the fast Fourier spectral method.

4 Density Dissipation

The scalar density field ρ can be updated for every time step using any desired advection method, for instance a semi-Lagrangian technique. Due to the discretized nature of the field, numerical instabilities tend to lead to the dissipation of densities, which in the simulation of liquids is seen as a loss of surface detail.

This problem can be solved by means of level set techniques [Fedkiw et al. 1999; Losasso et al. 2006]. However, these techniques are not fast enough for real-time applications owing to the need to

update the distance from the interface throughout the level set function. We therefore propose an alternative approach, which adapts the smoke-gathering technique developed by Fattal and Lischinski [2004]. In their technique, geometric shapes can attract smoke during the course of the simulation. This is achieved by creating a target density distribution ρ^* , and defining a gathering term $G(\rho, \rho^*)$ which concentrates smoke towards areas where the target density distribution is high:

$$G(\rho, \rho^*) = \nabla \cdot (\rho \tilde{\rho}^* \nabla (\rho - \rho^*)) \quad (12)$$

where $\tilde{\rho}^*$ is the target density distribution convolved with a Gaussian filter kernel. Updating the liquid density distribution is then given as an extension to (3):

$$\frac{\partial \rho}{\partial t} = -\mathbf{u} \cdot \nabla \rho + v G(\rho, \rho^*) \quad (13)$$

where v is a user-defined constant that determines the rate at which the gathering operates.

Since they show that this operator is mass conserving, we can employ this to ensure that the surface of the fluid remains well-defined. To this end, we compute ρ^* as follows. At the start of the simulation, we have a given amount of liquid, determined by the densities defined for each voxel. From this, we compute the average density over all non-zero voxels, which is subsequently used as a threshold T_ρ . The number of non-zero voxels c_T is counted as well. This number of voxels will be the target number of non-zero voxels which we aim to maintain during the simulation.

After each time-step, fluid advects across the volume, which means that the boundaries of the original fluid shift. If this means that the number of non-zero voxels has increased, suggesting dissipation has occurred, then we would like to reduce this number of voxels again, and vice-versa. A convenient way to achieve this is to rely on a statistical approach. Thus, our target density distribution is given by:

$$\rho^* = \begin{cases} \rho & \text{if } \rho \geq T_\rho \\ 0 & \text{if } \rho < T_\rho \end{cases} \quad (14)$$

We then update the aforementioned threshold after each time step, as follows:

$$T_\rho \leftarrow \frac{c_{\text{tot}} - c_T}{c_{\text{tot}} - c_v} T_\rho \quad (15)$$

where c_{tot} is total number of voxels in the liquid, and c_v is the number of voxels in the current time step that have a density ρ larger than T_ρ . This is the criterion we set for that voxel to belong to the body of liquid. This computation therefore either increases or decreases our threshold slightly.

Finally, we recompute $\tilde{\rho}^*$ from ρ^* . As this requires the computation of a Gaussian convolution, this is conveniently computed by a point-wise multiplication in the Fourier domain. To impose boundary conditions compatible with the mass conservation step, we use the 3D DCT-II domain \mathbf{P}^{CCC} where the filter kernel is precomputed using the DCT-I transform. Although we have transformed our filter kernel with a DCT-I transform, it can be proven that the resulting convolution with a DCT-II based transform is both efficient and in the DCT-II domain [Martucci 1994].

There are several advantages to preprocessing: in 3D the DCT-I implementation is a factor of eight slower than the DCT-II transform since the FFTW implementation does not have a fast and accurate implementation of the DCT-I whereas representing a Gaussian filter kernel by means of the DCT-II transform is awkward due to a half-sample offset.

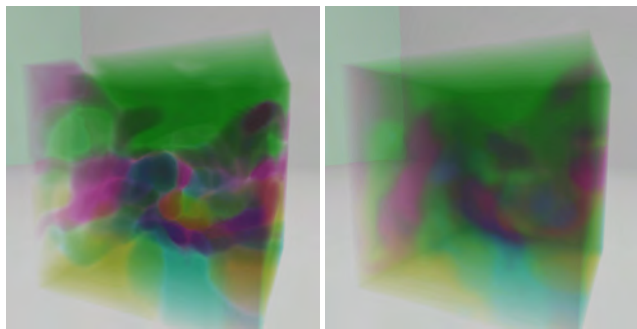


Figure 4: Images of an extended 64^3 simulation with and without gathering. See also Color Plate 3.

The reason that our target density distribution yields acceptable results is that the density advection dissipates density across the surface boundary of the previous time step, yielding a smoothly decreasing density distribution as function of distance to the surface. This means that iso-density surfaces at any suitably chosen value will yield a reasonable approximation to the surface of the fluid. This includes the iso-density surface at T_ρ .

The result is that over each frame the fluid becomes more like the target, which is also evolving due to the movement of the fluid. The fluid therefore moves correctly whilst detail is appropriately preserved. In Figure 4, we show that after a large number of solver steps, inevitable dissipation can be mitigated via our method. The compared images are simulated using the MacCormack method and monotone cubic Hermite spline interpolation as part of the advection step.

The main drawback of this approach is that groups of voxels that have zero fluid in them are unable to participate in the method. This can cause problems if a large volume of fluid is lost quickly and there are not enough non-zero voxels to regenerate it. The method currently does not support sub-voxel accuracy, a feature which would have to be implemented for physically accurate simulation. However, our algorithm is intended to be a computationally efficient substitute for level-set methods, and is applicable to applications for instance in entertainment.

5 Surface Tension

The realism of fluid simulations can be enhanced by modeling surface tension, creating effects such as the meniscus on a cup of water, or the shape of droplets. This tension is a force that depends on the molecules suspended in the fluid, and in terms of the Navier-Stokes equations, is folded into the \mathbf{f} component of (1).

Most practical implementations proceed in two steps. First, a surface boundary of the fluid is computed, followed by a computation of surface curvature. The amount of surface curvature then determines the surface tension. Surfaces can, for instance, be tracked using level set methods [Foster and Fedkiw 2001; Enright et al. 2002].

In level set methods, the position of a surface is indicated by a function ϕ , defined over the domain of the volume which evaluates to the distance to the interface. Its value will be zero at the surface, whereas $\phi > 0$ occurs inside the fluid, and $\phi < 0$ signifies parts of the volume which are outside the fluid.

In our method, we have density defined over the entire volume, which is intended to be 0 outside the volume, and $\rho > 0$ inside the volume. Apart from the signed distance metric required for a

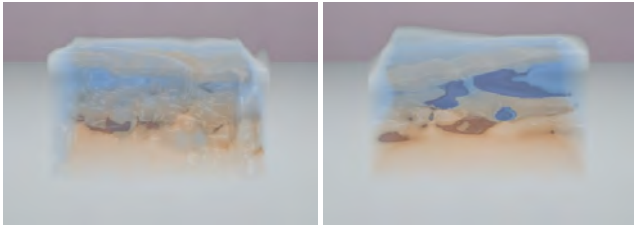


Figure 5: The left frame show our simulation without surface tension. The result is excessive mixing. The right shows liquids with surface tension applied.

true level set, ρ and ϕ serve the same purpose in surface tension computations. Thus, we follow Losasso et al [2006] and compute the surface curvature κ by:

$$\kappa = \nabla \cdot (\nabla \rho) = \frac{\partial^2 \rho}{\partial x^2} + \frac{\partial^2 \rho}{\partial y^2} + \frac{\partial^2 \rho}{\partial z^2} \quad (16)$$

However, rather than compute this value only at the boundary, we can compute this value throughout the volume. Noting that inside a liquid density gradients $\nabla \rho$ tend to zero, this would yield the correct result as κ will only be non-zero near the surface boundary. Given that κ requires the computation of the second spatial derivatives of ρ , we can efficiently compute κ in the DCT-II domain. We convolve ρ with a small Gaussian before computing κ . The Gaussian convolution serves to place an upper bound on the value of κ as well as to help prevent discretization artifacts. We can then calculate κ easily from second derivatives, as in DCT-II space these are simple scalings. The effect of applying surface tension is shown in Figure 5. The left frame is simulated without surface tension. The result is a rather muddled appearance, which can be significantly improved by adding surface tension, shown for the same simulation on the right.

6 Viscosity

Previously other Fourier spectral method implementations have computed viscosity in frequency space [Stam 1999]. This is computed via the evaluation of the viscosity term from the advection equation (1). Expanded in DST/DCT space (where \mathbf{U}' is defined as in (10)), this expression becomes:

$$-\pi \mu \left(\left(\frac{k+1}{K} \right)^2 + \left(\frac{l+1}{L} \right)^2 + \left(\frac{m+1}{M} \right)^2 \right) \mathbf{U}'(k, l, m) = \mu \mathbf{L} \quad (17)$$

where \mathbf{L} is the Fourier transform of the vector Laplacian. This has been tightly integrated into the mass conservation step for very little added cost, under the assumption that μ is globally constant. Instead if μ is spatially varying then following Durand and Dorsey [2002], we can split the range of viscosity coefficients into a number of bands, in which we assume the viscosity coefficient μ_i to be constant: $\sum_i \mu_i \nabla^2 \mathbf{u}$. As a result we can approximate (17) by piecewise linear interpolation of a set of DCT/DST transforms $\sum_i \mu_i \mathbf{L}_i$. We found that typically 2 transforms are required to give us sufficient control over the appearance of viscous fluids. An example of two fluids with different viscosities is shown in Figure 6.

7 Multiple Interacting Liquids

The above techniques can be extended to simulations of multiple liquids. For each liquid, we compute gathering and surface tension

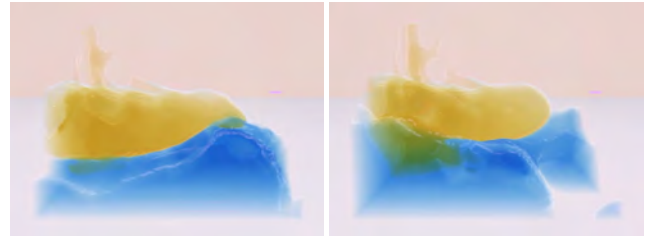


Figure 6: The yellow fluid has a much higher viscosity than the blue fluid, as shown in these two frames from the same animation.

separately. This does not affect the advection and mass conservation steps as all liquids share a common velocity field.

As a force produces an acceleration on a liquid which depends on its density, we would have to solve an additional mass conservation step ($\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0$), which contains a time-dependent and spatially varying density term $\frac{\partial \rho}{\partial t}$ [Guermont and Quartapelle 2000]. This term invalidates the conditions required for incompressible flow, making a real-time solution difficult. We therefore omit this equation, producing the Boussinesq approximation:

$$\rho \left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \right) = -\nabla p + \mu \nabla^2 \mathbf{u} + f \quad (18)$$

Thus, the internal forces between fluids of different density are approximated by (2), producing plausible results for non-critical applications such as entertainment, and in particular games.

8 Multi-Threading

The voxel domain is split into N slices, which are processed with P threads, such that N/P is an integral value. Voxels are laid out in memory such that the index of voxel (x, y, z) is given by $x + K(y + Lz)$. To maximize cache coherence, we therefore assign consecutive blocks of N/P slices to each thread, whereby the slices are aligned with the $x - y$ plane. All computations in the spatial domain are executed in multi-threaded mode on these slices.

As Equation (10) requires a combination of DCT and DST transforms, no efficient off-the-shelf implementations are directly available. However, with the FFTW codelet generator [Frigo 1999], we can construct an efficient dedicated transform. First we generate a one-dimensional standard real-to-real out of place DCT and IDCT. We then change the data type of the inputs and outputs, changing a float array input/output to a SIMD vector input/output. Given that a DCT has inputs and outputs $1 \dots N$, we can change a DCT into a DST by negating even inputs and inverting the order of the outputs [Shao and Johnson 2008]. The IDST is created from the IDCT in a similar fashion. Using the FFTW codelet generator in this manner doubles the efficiency of our combined DCT/DST transforms. Where we use DCTs or other transforms we do not use codelets, as the performance increase is not sufficient.

The computations in DCT/DST space are then carried out once more in multi-threaded fashion, using the same slicing scheme as outlined above. This scheme is efficient due to the fact that the DST and DCT transforms are real-to-real, and produce an equal number of coefficients as there are voxels.

9 Rendering

For the purpose of demonstrating our results, we use a GPU-based renderer that allows the ray equation to be directly evaluated. The

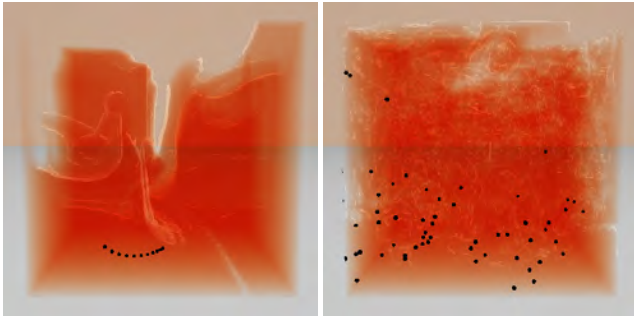


Figure 9: Frames 50 and 230 of a simulation releasing particles from the same position at a constant rate. The distribution of the particles shows the turbulence created in the fluid.

ray equation is a partial differential equation that models the curvature of a ray as it passes through a medium with a smoothly varying index of refraction [Stavroudis 1972]. Our implementation follows the approach described by Ihrke et al [2007]. The advantage is that this method can be tuned such that the voxel data produced by the simulation can be passed directly to the GPU for rendering, without the need for explicit surface extraction.

To derive an index of refraction for each voxel, we simply scale the density ρ to an appropriate range using $n = 1 + s\rho$, where s is a user-defined scaling factor whose purpose it is to give the index of refraction a realistic value. If ρ is around 1, then $s = 0.333$ would mean $n = 1.333$, the refractive index of water. In our implementation, the indices of refraction are spatially varying, as well as dynamic. This means, for instance, that we are able to render realistic mirages. In addition, using the ray equation does not preclude effects such as extinction using Beer’s law, so that we are able to accurately model light interaction with liquids

10 Results

Some of the visual effects achievable at real-time rates are shown in Figure 1. In this section, we show further examples, and present timing results. All our experiments were carried out using a quad-core 2.4 GHz Intel Core 2 processor with 2 GB RAM and an NVIDIA 8800 Ultra graphics card. Unless specified otherwise, the simulations run with a resolution of 64^3 voxels.

While most of our images show the behavior of liquids, the method can be adjusted to gaseous phenomena by omitting the gathering and surface tension steps. The simulation can then be tuned to take on the properties of gaseous phenomena such as fata morganas, mirages or even smoke as shown in Figure 7.

Our approach is capable of simulating free surfaces, as shown in Figure 1 (left), where in the starting condition a fluid is placed underneath a much denser transparent fluid. This gives rise to a Rayleigh-Taylor instability, shown here as a highly contorted surface. In Figure 8, we have created three liquids with different densities, with the densest at the top and the least dense at the bottom. This is also an unstable configuration, as a result of which the fluid mixes and then separates again into a stable configuration.

The inclusion of vorticity confinement helps create eddies and currents, shown in Figure 9 for a fluid of low viscosity and low density. Here, one particle is released every 4 frames, showing that as the simulation progresses, the entropy of the system increases. The liquid boundary is made reflective to aid this demonstration.

Finally, Figure 10 shows the effect of increasing the grid resolution

Voxels	1 Liquid		4 Liquids	
	Solver	Renderer	Solver	Renderer
32^3	96.33	26.39	58.56	17.88
64^3	14.29	13.73	8.01	8.67
128^3	1.63	6.28	0.87	4.01
256^3	0.19	0.78	-. ¹	-. ¹

¹ The graphics card does not have sufficient memory to render this scene.

Table 1: Performance statistics for both the simulation and the rendering algorithms. All numbers are in simulation timesteps per second for the solver or frames per second for the renderer.

from 32^3 to 256^3 voxels. The images are rendered at a resolution of 1024×768 . The ray tracing algorithm runs on the graphics card, and for those rays intersecting the volume, refractions are computed according to the ray equation. The resulting framerates are listed in Table 1. On current hardware, we see that a resolution of 64^3 leads to real-time performance, and 128^3 volumes can be simulated at interactive rates, both for single and multiple liquids.

A final observation is that if the starting configuration of the fluid is geometrically symmetric, then the simulation continues to generate symmetric distributions of density throughout the volume, as shown in Figure 10 (top). This result shows that our simulation is numerically stable. The bottom row is not symmetric, because the different liquids have different densities.

11 Conclusions

Boundary conditions are important in fluid simulations to obtain correct fluid flow. We have shown a fast approach to fluid simulation which includes boundary conditions. In addition, we explored fast solutions for density dissipation, surface tension and viscosity, and showed results for both liquids and gaseous phenomena. We are currently interested in extending the method to include arbitrary boundaries and obstacles.

We have shown that to compute a derivative, the DST domain can be used while maintaining proper boundaries, requiring no more than a simple scaling of the coefficients. To compute a second derivative, the DCT domain can be used in a similar manner. Using these basic tools, we have shown for the first time that real-time solutions with slip conditions at the data boundary are achievable by migrating to the DCT/DST domain. It should be a simple process to apply this approach to a pure DST domain to achieve no-slip boundaries.

Acknowledgments

We would like to thank Philip Child, and Tom Downes from Aardman Animation Studios as well as Timo Kunkel and Tania Pouli for their helpful insights and discussions.

References

- ANDERSON, J. D. 1995. *Computational fluid dynamics: The basics with applications*. McGraw-Hill, New York, NY.
- BATTY, C., BERTAILS, F., AND BRIDSON, R. 2007. A fast variational framework for accurate solid-fluid coupling. *ACM Transactions on Graphics* 26, 3, 100.
- BESSONOV, O., BRAILOVSKAYA, V., POLEZHAEV, V., AND ROUX, B. 1995. Parallelization of the solution of 3d navier-stokes equations for fluid flow in a cavity with moving covers.

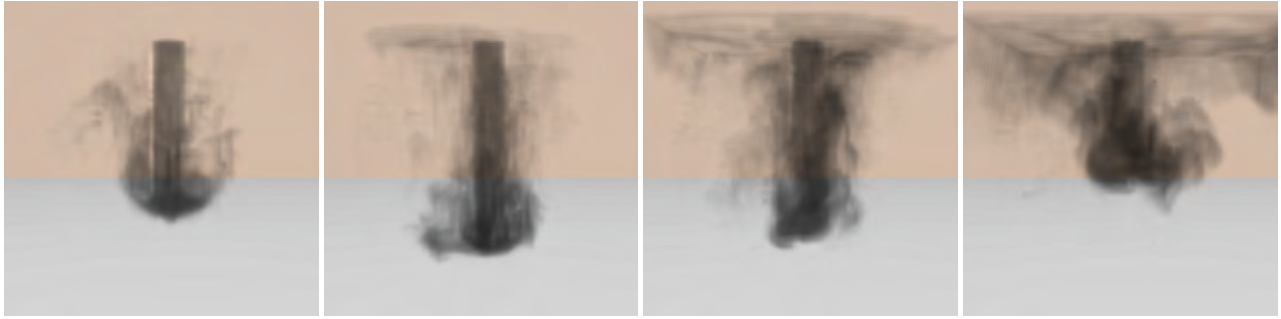


Figure 7: Our simulation can be configured to simulate smoke, which in this example is pushed into the system through a vent in the ceiling.

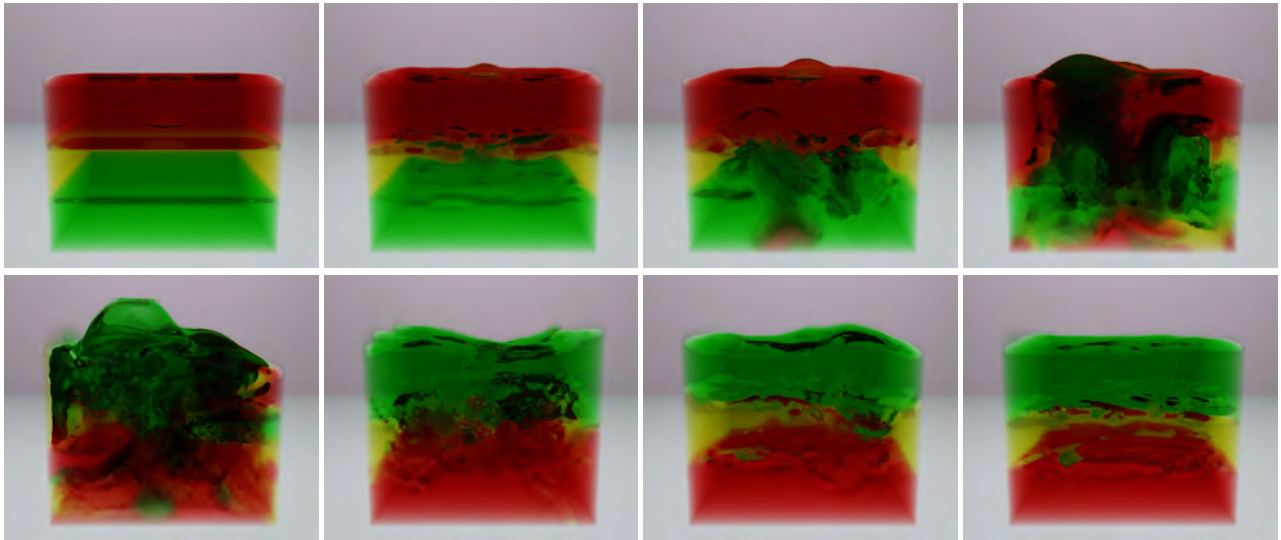


Figure 8: Simulation of three liquids with different densities. In the first frame, the highest density liquids are on top of the lower density liquids, as a result of which they mix and then separate again over time.

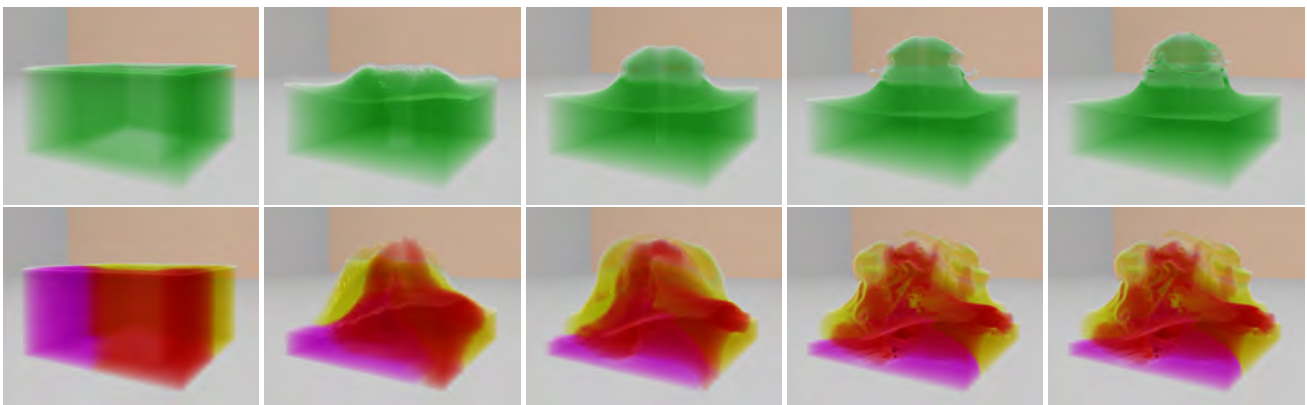


Figure 10: Simulation of four liquids at different grid resolutions. Advection is solved using the MacCormack method without vorticity confinement. Left to right: frame at $t = 0$. Frames at $t = 50$ using 32^3 , 64^3 , 128^3 , and 256^3 voxels. Top row: 1 liquid. Bottom row: 4 liquids.

- In *PaCT '95: Proc. of the 3rd International Conference on Parallel Computing Technologies*, 385–399.
- BOYD, J. 1999. *Chebyshev and Fourier spectral methods*, 2nd ed. Dover Publications, Mineola, NY.
- CARLSON, M., MUCHA, P. J., AND TURK, G. 2004. Rigid fluid: Animating the interplay between rigid bodies and fluid. *ACM Transactions on Graphics* 23, 3, 377–384.
- DENIS, B., CÔTÉ, J., AND LAPRISE, R. 2001. Spectral decomposition of two-dimensional atmospheric fields on limited-area domains using the discrete cosine transform (DCT). *Monthly Weather Review* 130, 1812–1829.
- DUPONT, T. F., AND LIU, Y. 2003. Back and forth error compensation and correction methods for removing errors introduced by uneven gradients of the level set function. *Journal of Computational Physics* 190, 1, 311–324.
- DURAND, F., AND DORSEY, J. 2002. Fast bilateral filtering for the display of high-dynamic-range images. *ACM Transactions on Graphics* 23, 3, 257–266.
- ENRIGHT, D., MARSCHNER, S., AND FEDKIW, R. 2002. Animation and rendering of complex water surfaces. *ACM Transactions on Graphics* 21, 3, 736–744.
- FATTAL, R., AND LISCHINSKI, D. 2004. Target-driven smoke animation. *ACM Transactions on Graphics* 23, 3, 441–448.
- FEDKIW, R., ASLAM, T., MERRIMAN, B., AND OSHER, S. 1999. A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method). *Journal of Computational Physics* 152, 457–492.
- FEDKIW, R., STAM, J., AND JENSEN, H. 2001. Visual simulation of smoke. In *Proceedings of ACM SIGGRAPH '01*, 15–22.
- FELDMAN, B. E., O'BRIEN, J. F., AND KLINGNER, B. M. 2005. Animating gases with hybrid meshes. *ACM Transactions on Graphics* 24, 3, 904–909.
- FOSTER, N., AND FEDKIW, R. 2001. Practical animation of liquids. In *Proceedings of ACM SIGGRAPH 2001*, 23–30.
- FOSTER, N., AND METAXAS, D. 1997. Modeling the motion of a hot, turbulent gas. In *Proceedings of SIGGRAPH '97*, 181–188.
- FRIGO, M., AND JOHNSON, S. G. 2005. The design and implementation of FFTW3. *Proceedings of the IEEE* 93, 2, 216–231.
- FRIGO, M. 1999. A fast fourier transform compiler. In *ACM SIGPLAN'99 Conference on Programming Language Design and Implementation (PLDI)*, 169–180.
- GÉNEVEAUX, O., HABIBI, A., AND DISCHLER, J.-M. 2003. Simulating fluid-solid interaction. In *Graphics Interface*, 31–38.
- GUENDELMAN, E., BRIDSON, R., AND FEDKIW, R. 2003. Non-convex rigid bodies with stacking. *ACM Transactions on Graphics* 22, 3, 871–878.
- GUERMOND, J.-L., AND QUARTAPELLE, L. 2000. A projection FEM for variable density incompressible flows. *J. Comput. Phys.* 165, 1, 167–188.
- HOCKNEY, R. W. 1965. A fast direct solution of poisson's equation using fourier analysis. *J. ACM* 12, 1, 95–113.
- IHRKE, I., ZIEGLER, G., TEVS, A., THEOBALT, C., MAGNOR, M., AND SEIDEL, H.-P. 2007. Eikonal rendering: Efficient light transport in refractive optics. *ACM Transactions on Graphics* 26, 3, 59.
- IRVING, G., GUENDELMAN, E., LOSASSO, F., AND FEDKIW, R. 2006. Efficient simulation of large bodies of water by coupling two and three dimensional techniques. *ACM Transactions on Graphics* 25, 3, 805–811.
- KAYIJA, J. T., AND VON HERZEN, B. 1984. Ray tracing volume densities. *Proc. of ACM SIGGRAPH 1984* 18, 3, 165–174.
- KIM, B. M., LIU, Y., LLAMAS, I., AND ROSIGNAC, J. 2005. Flowfixer: Using BFEC for fluid simulation. In *Eurographics Workshop on Natural Phenomena*, 51–56.
- KLINGNER, B. M., FELDMAN, B. E., CHENTANEZ, N., AND O'BRIEN, J. F. 2006. Fluid animation with dynamic meshes. *ACM Transactions on Graphics* 25, 3, 820–825.
- LAMORLETTE, A., AND FOSTER, N. 2002. Structural modeling of natural flames. *ACM Transactions on Graphics* 21, 3, 729–735.
- LOSASSO, F., SHINAR, T., AND SELLE, A. 2006. Multiple interacting fluids. *ACM Transactions on Graphics* 25, 3, 812–819.
- MAKHOUL, J. 1980. A fast cosine transform in one and two dimensions. *IEEE Transactions on Acoustics, Speech, and Signal Processing ASSP-28*, 1, 27–34.
- MARTUCCI, S. A. 1994. Symmetric convolution and the discrete sine and cosine transforms. *IEEE Transactions on Signal Processing* 42, 5, 1038–1051.
- NGUYEN, D., FEDKIW, R., AND JENSEN, H. 2002. Physically-based modeling and animation of fire. *ACM Transactions on Graphics* 29, 3, 721–728.
- REEVES, R., AND KUBIK, K. 2006. Shift, scaling and derivative properties of the discrete cosine transform. *Signal Processing* 86, 1597–1603.
- SELLE, A., RASMUSSEN, N., AND FEDKIW, R. 2005. A vortex particle method for smoke, water and explosions. *ACM Transactions on Graphics* 24, 3, 910–914.
- SELLE, A., FEDKIW, R., KIM, B. M., LIU, Y., AND ROSSIGNAC, J. 2007. An unconditionally stable MacCormack method. *Journal of Scientific Computing*, in review.
- SHAO, X., AND JOHNSON, S. G. 2008. Type-II/III DCT/DST algorithms with reduced number of arithmetic operations. *Signal Processing* 88, 6, 1553–1564.
- STAM, J. 1999. Stable fluids. In *Proceedings of ACM SIGGRAPH*, 121–128.
- STAM, J. 2001. A simple fluid solver based on the FFT. *Journal of Graphics Tools* 6, 2, 43–52.
- STAVROUDIS, O. N. 1972. *The Optics of Rays, Wavefronts and Caustics*. Academic Press, New York.
- STRANG, G. 1988. *Linear algebra and its applications*, 3rd ed. Harcourt College Publishers, Fort Worth.
- STRANG, G. 1999. The discrete cosine transform. *SIAM Review* 41, 1, 135–147.
- TREUILLE, A., LEWIS, A., AND POPOVIĆ, Z. 2006. Model reduction for real-time fluids. *ACM Transactions on Graphics* 25, 3, 826–834.
- YOKOKAWA, M., ITAKURA, K., UNO, A., ISHIHARA, T., AND KANEDA, Y. 2002. 16.4-Tflops direct numerical simulation of turbulence by a Fourier spectral method on the Earth Simulator. In *Supercomputing '02: Proceedings of the 2002 ACM/IEEE Conf. on Supercomputing*, 1–17.