Eurographics/ACM SIGGRAPH Symposium on Computer Animation (2004)
R. Boulic, D. K. Pai (Editors)

Directable Photorealistic Liquids

N. Rasmussen[1], D. Enright[2], D. Nguyen[3], S. Marino[4], N. Sumner[1], W. Geiger[1], S. Hoon[1], R. Fedkiw[3]

[1] Industrial Light + Magic, {nick, nsumner, wgeiger, samir}@ilm.com
[2] University of California, Los Angeles and Industrial Light + Magic, enright@math.ucla.edu
[3] Stanford University and Industrial Light + Magic, {dqnguyen, fedkiw}@stanford.edu
[4] Industrial Light + Magic, smarino@imageworks.com

Abstract

We present a method for the directable animation of photorealistic liquids using the particle level set method to obtain smooth, visually pleasing complex liquid surfaces. We also provide for a degree of control common to particle-only based simulation techniques. A variety of directable liquid primitive variables, including the isosurface value, velocity, and viscosity, can be set throughout the liquid. Interaction of thin liquid sheets with immersed rigid bodies is improved with newly proposed object-liquid boundary conditions. Efficient calculation of large-scale animations is supported via a multiple grid pipelined flow method and a novel moving grid windowing technique. In addition, we propose a few significant algorithmic enhancements to the basic liquid simulation algorithm to provide for the smooth merging of liquid drops, allow for the efficient calculation of high viscosity liquids, and ensure the proper treatment of isolated free liquid pockets surrounded by controlled liquid regions.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and RealismAnimation;

1. Introduction

Providing animators control over the behavior of a liquid interface when creating a directed performance has proven to be elusive. Natural liquids generally exhibit a characteristically "sloppy" behavior as discussed in [FM97a], which one would like to maintain while at the same time providing for explicit control in regions of interest. In the past, liquid actors such as those in 'The Abyss" and "Terminator 2: Judgment Day" have been created with realistic rendering, but without the desirable "sloppy" liquid behavior. In both cases, the liquid actor was directed via control vertices placed on the liquid surface. For complex large scale liquid animations, this approach quickly becomes infeasible. We instead propose a new method for the directable animation of photorealistic liquids as seen in figure 1. This character exhibits both controlled (e.g. the motion of the ridges around the nose) and uncontrolled (e.g. the dripping chin) liquid behavior.

Although we first reported on this control system over one year ago in [SHG*03], this paper provides many of the missing details. To our knowledge, this is the first liquid animation system that provides for such an extensive degree of control, while still remaining practical enough for a production environment. More recently, [MTPS04] proposed control techniques for liquids as well, but their results were limited to grids with two orders of magnitude less grid points than ours making their method impractical for a production environment.

Figure 1: *Use of "sloppy" liquid as a main actor.*

The particle level set method of [EMF02] is particularly effective in creating photorealistic behavior for complex, three dimensional liquids when combined with the basic Navier-Stokes solver of [FF01], which uses the "sta-

© The Eurographics Association 2004.

delivered by EUROGRAPHICS DIGITAL LIBRARY www.eg.org diglib.eg.org

ble fluids" semi-Lagrangian method of [Sta99] for efficiency. We extend the particle level set coupling technique of [EMF02] and control methodology introduced by [FM97a] and later [FF01] to create a method for the directable control of photorealistic liquids. Control particles are used to provide the desired degree of directed behavior. Associated with each particle is a control shape which defines the region of influence of the particle, and within this region a falloff curve is used to determine the amount of control applied to the liquid. A *soft* degree of control can be applied to any of the controllable liquid properties, e.g. viscosity. We also propose a *hard* control method for the explicit directed movement and visual appearance of the liquid interface. This allows for precise control over the behavior of the liquid, necessary for example when interacting with other animated characters or objects.

We introduce a variety of algorithmic improvements to the basic liquid animation technology of [FF01] to give robustness and flexibility to our method. We propose a new, loosely coupled multiple grid sourcing technique to allow for the calculation of large scale "down and out" liquid behavior without either a decrease in grid resolution or a significant increase in computational time over a single grid simulation. We propose a novel moving grid windowing method to track only the visually relevant portion of a liquid animation. A fast and stable method for variable viscosity liquids is also presented. In addition, we propose the use of a divergence free velocity extrapolation technique to facilitate the smooth merging of liquid masses. Object-liquid boundary conditions are improved to allow for the natural interaction of the particle level set representation of the liquid interface with rigid bodies. We also propose a simple flood fill technique to properly account for isolated pockets of free liquid surrounded by regions of hard control.

The signed distance from the interface allows for efficient ray tracing of the liquid. The liquid is rendered as an element in the final scene in order to capture the reflections of other objects in the liquid interface, and occlusions. Texture mapping of the liquid surface employed a combined particle and level set approach as well. Texture coordinates were attached to a set of particles which were passively advected along with the liquid interface. During the ray tracing process, texture coordinates of the particles nearest the intersection point of the ray with the liquid surface were blended together in order to define suitable texture coordinates. A simple look up procedure was used to calculate the texture coordinate at later times. Due to the advection of the particles through the liquid, the texture defined by the particles flows smoothly with the surface.

2. Previous Work

Various models have been proposed to capture the behavior of different types of liquid phenomena. Large scale 2D water surfaces have been represented by a variety of techniques ranging from spectral methods [MWM87, Tes02] to wave trains [FR86, Pea86] and hybrid spectral wave-train models [HNC02]. Liquid models more appropriate to a dynamic animation environment have utilized some form of the Navier-Stokes equations. [KM90] used a linearized form of the 2D shallow water equations to obtain a height field representation of the liquid surface. [CL94] and [OH95] combined interactivity with immersed objects and particle-based splashing respectively with a 2D height field fluid model. Recent SPH approximations to the Navier-Stokes equations include [PTB*03, MCG03].

[FM96] utilized the work of [HW65] in developing a grid based 3D Navier-Stokes methodology for the realistic animation of liquids. An unconditionally stable semi-Lagrangian treatment of the convective part of the Navier-Stokes equations was introduced to the computer graphics community in [Sta99]. [FF01] made significant contributions to the animation of three dimensional liquids through the introduction of a hybrid liquid volume model combining implicit surfaces and massless marker particles. [EMF02] demonstrated the particle level set method's ability to maintain visually pleasing thin sheets of water. Work incorporating bubbles, spray and foam can be seen in [HK03] and [TFK*03]. Additional research into object-liquid interaction has been done in [GHD03] which uses a point mass-spring object model and in [HBW03] and [WR03] which use level sets to model objects. Previous work in formulating fluid boundary conditions for animation purposes can be found in [FM97a] and [FM97b]. Recently, [LGF04] devised a symmetric pressure discretization on an unrestricted octree enabling an efficient adaptive approach to liquid simulation.

To complete the visual appearance of a liquid actor, texture mapping is required. Texture mapping of implicit surfaces has traditionally focussed on static surfaces, e.g. see [Ped95]. Two-dimensional advected textures for fluids were addressed in [Sta99, Wit99, Ney03]. Stunning imagery was created via flow on surfaces in [Sta03].

Natural phenomena have been used as an active dramatic element for visual effect purposes. Early use of particle based systems for fire effects can be seen in [Ree83] and continuing to the present with [LF02]. Particles can provide animators with the maximum degree of behavioral control, which is desirable when sculpting a performance. The use of velocity and force fields to control grid based liquid animations have been proposed in [Gat94, FM96, FM97a, WR03]. See also [SF93]. [TMPS03] used velocity fields in an automated key frame approach to control the "fuzzy" behavior of smoke. [FF01] introduced the idea of using point control on a liquid. Through this control technique, small liquid regions can be trapped and their behavior directed by an animator.

Most recently, [MTPS04] proposed a method for controlling liquids. Although they made large speed improvements over the work in [TMPS03], their coarse grids and large computational demands render this method unsuitable for a

© The Eurographics Association 2004.

production environment. While we readily simulate multiple 200^3 grids with an overnight computation, [MTPS04]'s examples ranged from 30^3 to $45x50x36$. This finest grid has 81,000 elements and took .6 GB of memory and 2 days to simulate. In contrast, our 200^3 grid has 8 million, or two orders of magnitude more elements. Even if their computations scaled linearly in the number of grid points (typical of a standard fluid simulation), it would take 60 GB of memory and 200 days to simulate a single 200^3 grid. Moreover, [MTPS04] uses a level set only technique employing artificial fluid sourcing to combat the significant mass loss incurred without using particles. This sourcing causes fluid to nonphysically disappear and reappear instead of convect around in a visually appealing fashion. In fact, the authors themselves dismiss this sourcing approach for smoke where they are able to avoid it.

3. Simulation Method

On each computational grid (our method allows for more than one), we solve the Navier-Stokes equations,

$$\nabla \cdot \mathbf{u} = 0, \tag{1}$$

$$\mathbf{u}_t = -\mathbf{u} \cdot \nabla \mathbf{u} + \nabla \cdot \tau - \frac{1}{\rho} \nabla p + \mathbf{g}, \tag{2}$$

where ρ is the density of the liquid, p is the pressure, $\mathbf{u}$ is the liquid velocity, $\mathbf{g}$ is an externally applied gravity field, and τ is the liquid viscous stress tensor. Each computational domain is divided into voxels with the components of $\mathbf{u}$ stored on the appropriate faces and p and ρ stored at the center of the cell. This arrangement of computational variables is the classic staggered MAC-style arrangement [HW65]. The approach of [FF01] is used to numerically solve for the various terms in equation 2 using the "stable fluids" semi-Lagrangian method of [Sta99] to update the convective terms. The discretization of the viscous force term, i.e. $\nabla \cdot \tau$, in equation 2 is discussed in section 5.1. To calculate the pressure in equation 2, the mass-preserving incompressibility constraint in equation 1 is used. The resulting linear system is solved with a fast, preconditioned conjugate gradient method. A constant pressure boundary condition is enforced in the "air" region outside the liquid. A modified version of the simple velocity extrapolation technique proposed in [EMF02] is used to satisfy incompressibility in cells near the interface as discussed in section 5.2.

The liquid volume is represented as an isocontour of an implicit function, ϕ. The surface of the liquid is defined by the $\phi = 0$ isocontour, the liquid by $\phi \leq 0$ and, the "air" by $\phi > 0$. This implicit surface description is passively evolved by the underlying liquid velocity field according to

$$\phi_t + \mathbf{u} \cdot \nabla \phi = 0. \tag{3}$$

A convenient representation of ϕ is as a signed distance function, making the accurate calculation of interface geometrical quantities straightforward, e.g. surface normals are given by $\mathbf{N} = \nabla\phi/|\nabla\phi|$. For a discussion of the numerical properties and various algorithms associated with ϕ, see [OF02]. To avoid excessive amounts of numerical diffusion when solving equation 3, the particle level set method of [EMF02] is used. Particles are placed in a band about the $\phi = 0$ isocontour. Particles in the unmodeled air region move according to the extrapolated velocity field, while particles inside the liquid use the underlying liquid velocity field $\mathbf{u}$. A newly proposed semi-Lagrangian treatment of the level set function in equation 3 combined with a fast second order Runge-Kutta time integration of the particles is used to avoid the computationally expensive HJ-WENO performed in [EMF02] without any significant degradation to the interface, see [ELF04].

Specialized boundary conditions are needed to account for the presence of objects in the liquid. The particle level set boundary conditions discussed in sections 5.5.2 and 5.5.3 are imposed during the advection phase for both the particles and level set. The object-velocity boundary conditions of section 5.5.1 are used when evolving the velocities forward in time according to equation 2. Application of the object-velocity boundary conditions are done in the same manner as proposed in [FF01].

4. Particle Based Controls

4.1. Basics

We define four types of control particles based upon the liquid variables we wish to control: velocity, viscosity, level set, and the velocity divergence. We use velocity particles to control the movement of the liquid interface, and level set particles to explicitly modify the level set representation of the liquid interface. Viscosity particles provide for visually pleasing melting behavior as seen in figure 3. Divergence particles enable the modeling of expansion/contraction of the liquid (see also [FOA03]). The region and amount of influence a control particle has is determined by its shape and falloff curve as discussed below.

Spline-based interpolation of keyframed position and velocity values is used to determine particle positions during the entire animation. All of the particle attributes are determined prior to animating the liquid, although it is possible to stop an animation in midstream and change the various control particle attributes if desired. Through the extensive use of our animation system, we have found that a particle density of one control particle per grid cell allows for a sufficient degree of control over an animation without too much over or under sampling. Keyframing of control particles is easily accomplished with a standard production animation package, and we implemented this aspect of our liquid animation system in Maya as shown in figure 2.

4.2. Control Shapes and Falloff Curves

A control particle's region of influence is determined by its shape. We primarily use two basic shapes: a sphere for

© The Eurographics Association 2004.

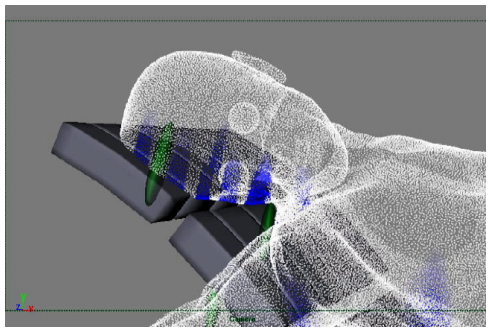


Figure 2: *Control particles (in white) applying a hard velocity control. Blue regions have soft velocity control and green regions are CSG level set liquid erasers.*

Figure 3: *Melting and flowing of a liquid metal creature.*

isotropic control and a cylinder for anisotropic control. A spherical control shape is normally used when manipulating velocity and viscosity fields. A cylindrical control shape is used when directing the level set function, since it provides the ability to set the surface normal to a desired direction. The typical radius of a control shape is between one-half to one grid cell in size.

The amount of control applied to the grid cells within the region of influence of a particle is determined by an animator set falloff curve. A box function, a hat function, and a smooth cubic interpolating function are typically used. Each of these functions are scaled to vary between zero and one, with the function set to one at the center of the region. For a spherical control region, the falloff curve is a function of the distance to the center of the sphere, while for a cylindrical control region it is a function of the distance to the major axis through the center of the cylinder. A box function is often used when we want ultimate directorial control, and a hat function provides for continuity at the boundary of the region of influence. For smoother control behavior, a cubic interpolating function with zero derivative at the end points is used. This function is also continuous at the boundary of the control region.

4.3. Soft and Hard Control

After defining all of the attributes of the control particles, the exported particle information is incorporated into the liquid simulation engine. From the control region of each particle, the grid cells that will undergo directorial control can be determined. A "soft" or blended control is typically used. The amount of mixing between the animator desired value, $V_{particle}$, and the liquid value currently used by the simulation, V_{liquid}, is given by the falloff curve value α in each cell. A convex combination of these two values is formed to determine the value used by the simulation, i.e.

$$V_{control} = (1-\alpha)V_{liquid} + \alpha V_{particle}. \quad (4)$$

If a cell is undergoing control by N particles on the same liquid property, a weighted average is taken.

To control the exact timing and placement of the liquid interface, the velocity field may require a greater degree of control, i.e. a "hard" control. This is due to the imprecise amount of control provided by blended force and wind fields whose effect is blurred by the inherent nonlinear velocity convection present and the global coupling of the velocity field through the pressure term. A hard control over a region of liquid is obtained by fixing, independent of the physics-based model, the velocity field inside the region undergoing hard control and enforcing a Neumann type pressure boundary condition at the boundary of this region. Then the pressure solver naturally provides us with a smoothly varying velocity field away from the controlled region and into the surrounding uncontrolled liquid. We enforce a hard control on a cell if $\alpha > .9$. Figure 2 illustrates control particles in white applying a hard control to the liquid metal surface of an animated creature. The final rendered result is shown in figure 3.

While applying a hard control to the interface produces the visual behavior we desire, sometimes grid based aliasing artifacts are encountered. For the liquid metal face seen in figure 1, the regions undergoing hard control, e.g. the ridges above the eyes, exhibited an excessive amount of smoothing when the movement of the liquid interface was determined solely by fixed velocity fields. To correct this aliasing artifact, when a hard control on the velocity field was imposed at this location we also imposed a soft control on the level set function on the same region via an additional set of level set control particles. Cylindrical control particles centered at the same position as the velocity control particles were used to achieve this. The animator set the desired normal direction for the liquid interface in the ridges above the eyes. This normal information along with the position of the control particle, defines a cross sectional plane to the cylinder which was used to calculate a $\phi_{particle}$ at each grid point inside the cylinder. This value is then blended with ϕ_{liquid} to retain the sharp eyebrow ridges. Since the interface position is being artificially controlled, any particles associated with the particle level set representation of the interface in this region are reseeded to the adjusted interface location. Through this process we are able to maintain a visually pleasing smooth liq-

© The Eurographics Association 2004.

uid interface everywhere, including regions undergoing animator directed control.

We also note that the volumetric description of the liquid interface lends itself to CSG style manipulations when a coarser degree of control is desired as seen in figure 2.

5. Algorithmic Improvements

5.1. Variable Viscosity

To enhance control we allow the viscosity of the material to vary in space and time. This variation in the viscosity can either be controlled by an animator using the methods discussed above or be made a function of another modeled field, such as temperature, as was done in [CMVHT02]. High viscosity liquids have been animated using a backwards Euler discretization [CMVHT02, FR03] in order to avoid any numerical instabilities associated with taking a large semi-Lagrangian time step. We also note that Stam's FFT-based solver for the viscous diffusion terms [Sta99] is equivalent to a backward Euler implementation in the case of constant viscosity when there is no free boundary present in the flow.

For the variable viscosity case, the viscous forcing terms resulting from the divergence of the viscous stress tensor, i.e. $\mathbf{F}_{visc} = \nabla \cdot \tau$, are given by (see e.g. [Pan96]),

$$F^x = (\nu(2u_x - \frac{2}{3}\nabla\cdot\mathbf{u}))_x + (\nu(u_y + v_x))_y + (\nu(w_x + u_z))_z, \quad (5)$$

$$F^y = (\nu(u_y + v_x))_x + (\nu(2v_y - \frac{2}{3}\nabla\cdot\mathbf{u}))_y + (\nu(v_z + w_y))_z, \quad (6)$$

$$F^z = (\nu(w_x + u_z))_x + (\nu(v_z + w_y))_y + (\nu(2w_z - \frac{2}{3}\nabla\cdot\mathbf{u}))_z, \quad (7)$$

where $\mathbf{u} = (u, v, w)$ and ν is the kinematic viscosity. For incompressible flow, the $\nabla\cdot\mathbf{u}$ terms vanish. A straightforward implicit discretization of equations 5-7 results in a matrix which is nonsymmetric and therefore expensive to invert. Instead, we propose an explicit-implicit splitting using a fast preconditioned conjugate gradient solver.

When advancing the velocity forward in time via equation 2, we first add in the contributions from the convective and body forcing terms to $\mathbf{u}^n$, obtaining $\mathbf{u}^*$. Next, we explicitly add the asymmetric components of equations 5-7 plus one of the two symmetric components, i.e.

$$u^* \mathrel{+}= \Delta t \left[(\nu u_x^n)_x + (\nu v_x^n)_y + (\nu w_x^n)_z \right], \quad (8)$$

$$v^* \mathrel{+}= \Delta t \left[(\nu u_y^n)_x + (\nu v_y^n)_y + (\nu w_y^n)_z \right], \quad (9)$$

$$w^* \mathrel{+}= \Delta t \left[(\nu u_z^n)_x + (\nu v_z^n)_y + (\nu w_z^n)_z \right], \quad (10)$$

where standard second order central differencing is used along with linear interpolation to obtain viscosity values where needed.

We now are left with the following systems of equations

Figure 4: *Melting wax mannequin head (left). Liquid flowing down a pirate skeleton's chest (right).*

to invert,

$$(I - \Delta t A_u)u^{**} = u^*, \quad (11)$$

$$(I - \Delta t A_v)v^{**} = v^*, \quad (12)$$

$$(I - \Delta t A_w)w^{**} = w^*, \quad (13)$$

where the A_i are symmetric constant coefficient matrices defined via,

$$A_u u^{**} = (\nu u_x^{**})_x + (\nu u_y^{**})_y + (\nu u_z^{**})_z, \quad (14)$$

$$A_v v^{**} = (\nu v_x^{**})_x + (\nu v_y^{**})_y + (\nu v_z^{**})_z, \quad (15)$$

$$A_w w^{**} = (\nu w_x^{**})_x + (\nu w_y^{**})_y + (\nu w_z^{**})_z. \quad (16)$$

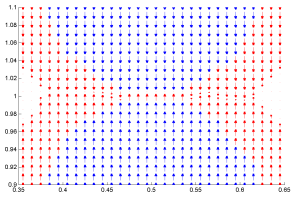
Standard central differencing is used to compute the derivatives, and ν is interpolated when necessary. Finally, we solve for the pressure and project out any non-divergence free component in $\mathbf{u}^{**}$ to obtain $\mathbf{u}^{n+1}$.

Use of this explicit-implicit viscosity treatment can be seen in figures 3 and 4 (left). If the viscosity is constant, the explicit part of the viscous terms shown in equations 8-10 are zero due to incompressibility. However, [CMVHT02] inaccurately neglects these in the variable viscosity case as well. (We also note that the artificial dissipation of high viscosity liquids in free flight that was noted in [CMVHT02] is probably due to a bug in their implementation. There is no null space in these equations.)

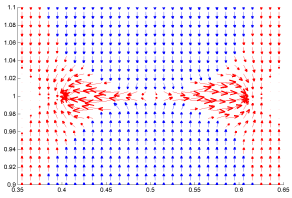
5.2. Divergence Free Velocity Extrapolation

While using the velocity extrapolation approach advocated by [EMF02], we observed that the method may sometimes have difficulty dealing with merging liquid interfaces. As shown in figure 5, two liquid drops are about to merge and their extrapolated velocity fields overlap due to the drops' proximity to each other. The behavior of the extrapolated velocity field and the "air" particles, both in red, can be seen in figures 5(a) and 5(c). The air particles between the two drops remain clustered between them due to the fact that the extrapolated velocity field, while satisfying incompressibility *under grid refinement*, fails to do so on the under-resolved grid seen here. This failure to satisfy the incompressibility

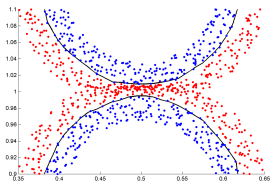
© The Eurographics Association 2004.



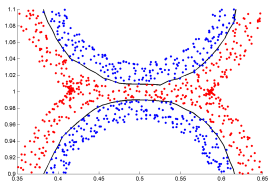
(a) Velocity - Not Divergence Free



(b) Velocity - Divergence Free



(c) Particle Distribution, Not Divergence Free



(d) Particle Distribution, Divergence Free

Figure 5: *Velocity Extrapolation - Two Colliding Drops*

constraint can result in the liquid and air mixing together to form a visually displeasing liquid interface and the possible loss of liquid mass.

To prevent this phenomenon from occurring, after extrapolating the velocity away from the interface we then also project out any non-divergence free component of the extrapolated velocity field. The resulting velocity field is everywhere discretely mass conserving, even in the unmodeled air region near the interface. The method used for the divergence free projection of the extrapolated velocity field is the same as for the liquid velocities. A constant pressure boundary condition is maintained in the air outside the extrapolated velocity region and a fixed velocity boundary condition is applied to the liquid. This divergence free velocity extrapolation technique was first proposed in [Sus03]. The resulting velocity field can be seen in figure 5(b) and the corresponding behavior of the air particles is shown in figure 5(d). The extrapolated velocity now behaves correctly, i.e. the air sees that it is being squeezed by the drops and forms two high speed jets, one to the right the other to the left, to relieve the pressure. The air particle density, while not zero, is substantially less than in figure 5(c), resulting in a smoother merging of the two drops. Note that if air particles do get trapped in a merging region, they are deleted if they penetrate too far into the liquid (as suggested in [ENGF03]).

5.3. Grid Windowing And Resizing

Capturing the detailed motion of liquids or liquid-like creatures interacting with other objects using volumetric simulation techniques can result in long computational times and large memory costs. To minimize these costs, a moving grid

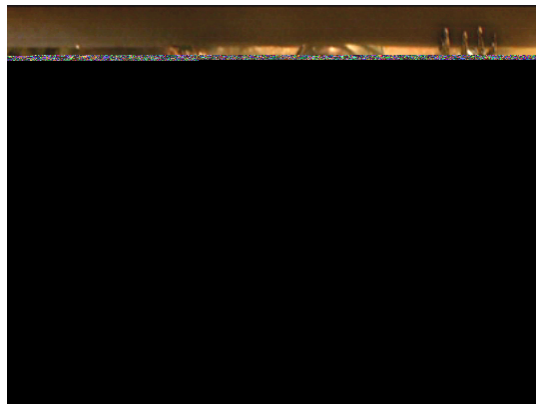


Figure 6: *Dynamically resized grid to capture dripping liquid metal train.*

windowing technique has been developed to restrict the simulation of the liquid to a region of interest as shown in figure 1 where liquid metal has been placed over an animated head mesh collision object. A simple approach for this simulation would have been to use a grid large enough to contain the entire movement of the head mesh. Instead, a much smaller moving grid window of the larger logical grid containing the entire movement of the head was used.

To accomplish this, the liquid primitive variables, ϕ, $\mathbf{u}$, and ν, are translated in their arrays to match the movement of the grid window. Also, the information describing the spatial location of each grid cell is changed accordingly. We restrict translations to whole grid cell increments to avoid any interpolation issues. As an example, if the grid window is translated to the left by three grid cells, the following data movement for the level set function is performed, $\phi_{i,j,k} \leftarrow \phi_{i+3,j,k}$ for $i = 1 \ldots N-3$, $j = 1 \ldots N$, and $k = 1 \ldots N$, where N is the number of grid cells per dimension. In the case of the head mesh simulation shown in figure 1, the entire head mesh was always centered in the middle of the moving grid window as the head moved from side-to-side. Thus, any uninitialized values resulting from the translated grid window were treated as unmodeled air values. To capture the liquid metal train dripping from the head in figure 6, we dynamically resized the grid window without incurring too much additional computational overhead.

We wish to emphasize that the process described above does not require any additional forcing terms to be added to equation 2 since we are not simulating the liquid in a non-inertial reference frame. Rather, we are just taking a window of the fixed logical grid in which the entire scene is taking place.

5.4. Grid Sourcing

While grid windowing can provide for higher resolution single grid liquid simulations, the animation of large scale liquid streams is still problematic due to the size of the grid necessary to obtain reasonable surface resolution. For freely

© The Eurographics Association 2004.

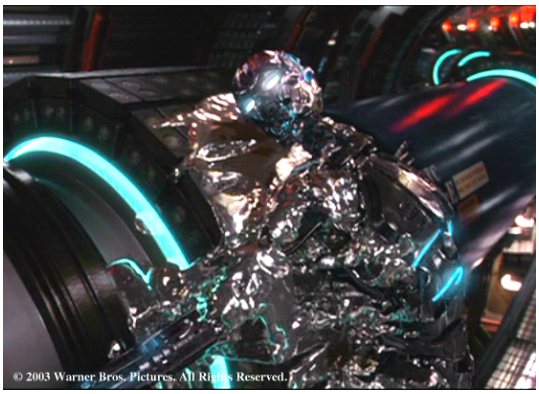


Figure 7: *Sourcing of liquid from the torso and down the arm of the creature.*

flowing gravity driven motion, i.e. liquid flowing down and out, liquid pressure effects are relatively unimportant to the behavior of the interface. Instead, the bulk convection of the liquid determines the behavior of the liquid interface. The proposed grid sourcing technique takes advantage of this fact by splitting up the overall liquid computational domain into smaller, higher resolution volumes that are loosely coupled together through the primitive liquid variables.

We "source", i.e. transfer, the liquid primitive variables from one grid to the next at every time step. The grids, which cover the domain of interest, are arranged in an acyclic directed graph indicating the flow direction. The grids slightly overlap each other in three grid cell deep bands to allow for the smooth transfer of data from one grid to the next. For grids lower in the dependency graph, the values sourced onto these grid are held constant when solving equations 1 and 2. The sourced boundary bands are stored to disk after each time step for convenient access. The sourcing of the liquid primitive variables allows the entire flow calculation to be pipelined, with grid volumes higher up the dependency graph continuing further on in the flow calculation without waiting for the volumes lower in the dependency graph to finish. By pipelining the calculation, a substantial savings in the overall computational time can be obtained. One difficulty encountered when pipelining the calculation with time asynchronous fluid volumes is that a volume further down the pipeline may demand sourced boundary information at a time level not explicitly computed by the volume upstream to it. In this case, we just interpolate the sourced data from the upstream volume to the appropriate time level.

An example of grid sourcing can be seen in figure 7, where flowing liquid metal from the melting torso and down the arm of the creature is captured with two computational volumes. An additional benefit of grid sourcing is the flexibility of adding additional computational volumes without having to restart the entire calculation. Unexpected flowing of liquid off a computational domain can be captured through the addition of a downstream computational volume using the stored sourced variables as the boundary conditions for the newly added volume. The upstream sourcing volume is able to finish its calculation normally.

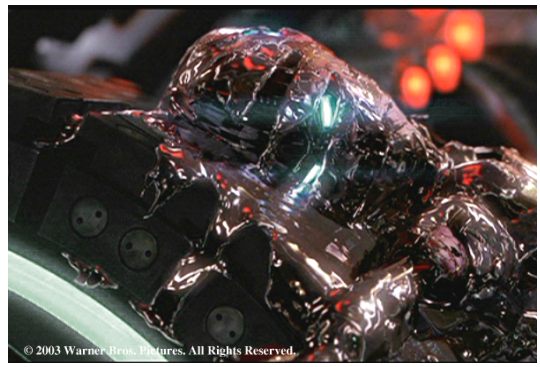


Figure 8: *Thin liquid sheets flowing from the melting creature.*

5.5. Boundary Conditions

Specialized boundary conditions are needed to ensure a visually pleasing interaction of the liquid interface with immersed objects. A common approach to incorporating objects into liquid animations is to enforce a velocity boundary condition such that no liquid is allowed to enter an object [FM96, FF01]. While relatively easy to implement, use of this method alone to ensure visually pleasing liquid-object interaction is challenging when using a grid based interface method. We instead propose a set of object boundary conditions for the velocity field and both the level set function and particles comprising the particle level set method. These boundary conditions are used for both moving and stationary objects as seen in figures 4 (right) and 8.

Although objects in our liquid animation method are intrinsically represented as triangulated rigid and deformable bodies, we use a corresponding volumetric representation on the liquid computational domain to allow the object to interact with the liquid. We maintain both an explicit and implicit representation of an object as in [GBF03]. The implicit representation, ϕ_{obj}, is constructed each time step from the animator-controlled explicit representation.

5.5.1. Velocity

We use the constrained velocity extrapolation approach of [HBW03]. First, we extrapolate the velocity field nearby the object into the object using ϕ_{obj}. Then for each grid cell within the object, a convex combination of the smoothly extrapolated velocity $\mathbf{u}_{ext}$ and the desired animator-controlled object velocity $\mathbf{u}_{obj}$ is taken, i.e. $\mathbf{u}_{comb} = (1-\alpha)\mathbf{u}_{ext} + \alpha\mathbf{u}_{obj}$ where $\alpha \in [0,1]$ is based upon the amount of object friction with the liquid. From $\mathbf{u}_{comb}$, we fix the tangential component of the velocity field inside the object to be $\mathbf{u}^{\parallel}_{fix} = \mathbf{u}_{comb} - (\mathbf{u}_{comb} \cdot \mathbf{N}_{obj})\mathbf{N}_{obj}$. For the normal velocity component inside the object, we enforce the condition that no liquid is allowed to flow into the object. We first calculate

© The Eurographics Association 2004.

$u^{\perp}_{rel} = (\mathbf{u}_{ext} - \mathbf{u}_{obj}) \cdot \mathbf{N}_{obj}$. If the relative velocity indicates flow into the object ($u^{\perp}_{rel} < 0$), we set $u^{\perp}_{fix} = u^{\perp}_{obj}$, otherwise we set $u^{\perp}_{fix} = u^{\perp}_{ext}$. A hard control on the velocity cells inside an object is now set to ensure that the fixed velocity field is faithfully preserved during the pressure update as discussed in section 3.

5.5.2. Level Set

An obvious representation of an object would be to assign the interior of the object a value of $\phi > 0$. However, this condition can pose aliasing issues when solving equation 3, with the interior of an object incorrectly "leaking" air into the liquid. Thus, to set the ϕ values for cells inside the object, we extrapolate inwards the ϕ values surrounding the object. By having a more consistent representation of ϕ both inside and outside the object, visual discrepancies resulting from equation 3 are minimized. However, with only this boundary condition for ϕ, it is difficult to cause sheets of liquid to separate from the object and into the air. Objects will now instead leak liquid. Therefore, while we extrapolate ϕ into the object after solving equation 3, before solving equation 3 we correct ϕ inside the object in the following manner. First, we determine the relative velocity, i.e. $\mathbf{u}_{rel} = \mathbf{u}_{ext} - \mathbf{u}_{obj}$, for each cell inside the object. If $\mathbf{u}_{rel} \cdot \mathbf{N}_{obj} > .1|\mathbf{u}_{rel}|$ and $\phi < 0$ for a given cell, we reset the cell's ϕ value to be air (i.e. we set it to $|\phi_{obj}|$). We have found that this threshold condition produces satisfactory results allowing liquid to separate from the object.

5.5.3. Particles

"Air" particles do not collide with an object, and they are deleted if they drift more than a small distance inside the object. In contrast, "liquid" particles collide with the object. Standard collision treatment would project all the particles to the surface of the object causing them to pile up in an infinitesimally thin band, which doesn't provide enough thickness to resolve a thin film. This is similar to the cloth flattening problems discussed in [BMF03] and we use a similar solution. First, before updating the position of any particles, we label those that are in close proximity to the object (within about one grid cell) based on their ϕ values. After the positions of these particles are updated, we use the object normal to project these particles away from the object to ensure that they are no closer than their original ϕ value would imply. This keeps the particles in the band well stratified so that they can resolve thin liquid films. We additionally adjust the velocity of these particles to remove any normal component that would cause the particle to drift closer to the object.

5.6. Enslaving Isolated Free Liquid Pockets

Creation of isolated uncontrolled liquid pockets that are completely surrounded by regions of hard control can occur in many places in our directable liquid method, e.g. during grid sourcing or when placing complicated collision objects in the liquid. Calculating the pressure with one of the standard solver techniques will fail, since there is no guarantee that the boundary conditions do not force a violation of incompressibility. Thus, special treatment is needed for any liquid region which is completely surrounded by Neumann boundary conditions. One remedy would be to enforce a compatibility condition guaranteeing that incompressibility can be enforced. However, since the behavior of the isolated pockets of free liquid is completely forced due to the surrounding Neumann boundary conditions, we "enslave" the velocity field inside the pockets instead. During each time step we scan all the cells containing liquid to see if any isolated free fluid pockets exist. This can be efficiently done with a simple flood fill algorithm. The terminating condition for the flood fill recursion is when either a hard controlled boundary cell (which includes objects, walls, and/or fluid regions) or an air boundary cell is reached. If an air boundary cell is reached, the liquid region is not isolated and we ignore it. Otherwise we proceed as described below.

The velocity of each identified isolated pocket of free liquid can be set by a velocity extrapolation technique, i.e. extrapolating the boundary velocities inward. We instead use an even simpler diffusion technique iterating through the enslaved region a couple of times updating the velocity value of each enslaved cell with the average value of its neighbors. Either technique yields a smooth movement of the liquid in the enslaved region. In addition, by enforcing a fixed velocity on all the newly enslaved cells, these cells can be removed from the pressure solver, providing additional computational savings when calculating the pressure in the remaining free liquid cells.

6. Rendering

We use a combined particle and level set approach to texture map liquids as seen in figure 3. While the signed distance information of the level set is used for efficient ray-liquid intersection calculations, particles advected through the flow determine the surface texture. We place particles on the initial liquid surface which was made to correspond to a previously photographed background plate of the actor. To determine where in the background plate a particle should determine its color, first the particle position is transformed into normalized device coordinates. From the normalized device coordinates of a particle, the actual location on the plate to perform the texture lookup can be determined. After executing the liquid animation, the texture particles are then advected through the stored liquid velocity field. At a ray intersection point on the surface of the liquid, the initial positions of the closest 64 particles are interpolated to determine which coordinate to perform the aforementioned texture lookup procedure on. This interpolation procedure, when combined with the particle advection through the precomputed flow field, provides for a smooth flowing texture to the surface. While this procedure is sufficient to provide texturing to a flowing

© The Eurographics Association 2004.

surface from a static initial plate, if the texture in the background plate is also moving, then an additional time projection step is necessary. After determining the position on the initial surface to perform the look up, this position is projected onto the time correct surface from which the standard texture lookup procedure is performed. The projection is determined by the calculated velocity of the figure according to the background plates.

7. Conclusions

The two types of control discussed (i.e. "soft" or blended control of the primitive liquid variables and the explicit "hard" velocity control of the liquid interface), when combined with the underlying liquid animation method of Foster and Fedkiw [FF01] (enhanced with our algorithmic improvements), provides for the desirable procedural animation of "sloppy" liquid behavior.

8. Acknowledgement

Research supported in part by an ONR YIP award and a PECASE award (ONR N00014-01-1-0620), a Packard Foundation Fellowship, a Sloan Research Fellowship, ONR N00014-03-1-0071, ONR N00014-02-1-0720, NSF DMS-0106694, NSF ITR-0121288, NSF IIS-0326388 and NSF ACI-0323866. In addition, D. E. was supported in part by an NSF postdoctoral fellowship (NSF DMS-0202459). We would like to thank Cliff Plumer, Steve Sullivan and Industrial Light + Magic for their support and enthusiasm.

References

[BMF03] BRIDSON R., MARINO S., FEDKIW R.: Simulation of clothing with folds and wrinkles. In *Proc. of the 2003 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.* (2003), pp. 28–36. 8

[CL94] CHEN J., LOBO N.: Toward interactive-rate simulation of fluids with moving obstacles using the navier-stokes equations. *Computer Graphics and Image Processing 57* (1994), 107–116. 2

[CMVHT02] CARLSON M., MUCHA P., VAN HORN R., TURK G.: Melting and flowing. In *ACM SIGGRAPH Symposium on Computer Animation* (2002), pp. 167–174. 5

[ELF04] ENRIGHT D., LOSASSO F., FEDKIW R.: A fast and accurate semi-Lagrangian particle level set method. *Computers and Structures, (in press)* (2004). 3

[EMF02] ENRIGHT D., MARSCHNER S., FEDKIW R.: Animation and rendering of complex water surfaces. *ACM Trans. Graph. (SIGGRAPH Proc.) 21*, 3 (2002), 736–744. 1, 2, 3, 5

[ENGF03] ENRIGHT D., NGUYEN D., GIBOU F., FEDKIW R.: Using the particle level set method and a second order accurate pressure boundary condition for free surface flows. In *Proc. 4th ASME-JSME Joint Fluids Eng. Conf.* (2003), no. FEDSM2003–45144, ASME. 6

[FF01] FOSTER N., FEDKIW R.: Practical animation of liquids. In *Proc. of ACM SIGGRAPH 2001* (2001), pp. 23–30. 1, 2, 3, 7, 9

[FM96] FOSTER N., METAXAS D.: Realistic animation of liquids. *Graph. Models and Image Processing 58* (1996), 471–483. 2, 7

[FM97a] FOSTER N., METAXAS D.: Controlling fluid animation. In *Computer Graphics International 1997* (1997), pp. 178–188. 1, 2

[FM97b] FOSTER N., METAXAS D.: Modeling the motion of a hot, turbulent gas. In *Proc. of SIGGRAPH 97* (1997), pp. 181–188. 2

[FOA03] FELDMAN B. E., O'BRIEN J. F., ARIKAN O.: Animating suspended particle explosions. *ACM Trans. Graph. (SIGGRAPH Proc.) 22*, 3 (2003), 708–715. 3

[FR86] FOURNIER A., REEVES W. T.: A simple model of ocean waves. In *Computer Graphics (Proc. of SIGGRAPH 86)* (1986), vol. 20, pp. 75–84. 2

[FR03] FÄLT H., ROBLE D.: Fluids with extreme viscosity. In *SIGGRAPH 2003 Sketches & Applications* (2003), ACM Press. 5

[Gat94] GATES W.: *Interactive Flow Field Modeling for the Design and Control of Fluid Motion in Computer Animation*. Master's thesis, University of British Columbia, 1994. Dept. of Computer Science. 2

[GBF03] GUENDELMAN E., BRIDSON R., FEDKIW R.: Nonconvex rigid bodies with stacking. *ACM Trans. Graph. (SIGGRAPH Proc.) 22*, 3 (2003), 871–878. 7

[GHD03] GÉNEVAUX O., HABIBI A., DISCHLER J.-M.: Simulating fluid–solid interaction. In *Graphics Interface* (2003), A K Peters, pp. 31–38. 2

[HBW03] HOUSTON B., BOND C., WIEBE M.: A unified approach for modeling complex occulsions in fluid simulations. In *SIGGRAPH 2003 Sketches & Applications* (2003), ACM Press. 2, 7

[HK03] HONG J.-M., KIM C.-H.: Animation of bubbles in liquid. *Comp. Graph. Forum (Eurographics Proc.) 22*, 3 (2003), 253–262. 2

© The Eurographics Association 2004.

[HNC02] HINSINGER D., NEYRET F., CANI M.-P.: Interactive animation of ocean waves. In *ACM SIGGRAPH Symposium on Computer Animation* (2002), pp. 161–166. 2

[HW65] HARLOW F., WELCH J.: Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface. *Phys. Fluids 8* (1965), 2182–2189. 2, 3

[KM90] KASS M., MILLER G.: Rapid, stable fluid dynamics for computer graphics. In *Computer Graphics (Proc. of SIGGRAPH 90)* (1990), vol. 24, pp. 49–57. 2

[LF02] LAMORLETTE A., FOSTER N.: Structural modeling of natural flames. *ACM Trans. Graph. (SIGGRAPH Proc.) 21*, 3 (2002), 729–735. 2

[LGF04] LOSASSO F., GIBOU F., FEDKIW R.: Simulating water and smoke with an octree data structure. *ACM Trans. Graph. (SIGGRAPH Proc.)* (2004). 2

[MCG03] MÜLLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. In *Proc. of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2003), pp. 154–159. 2

[MTPS04] MCNAMARA A., TREUILLE A., POPOVIĆ Z., STAM J.: Fluid control using the adjoint method. *ACM Trans. Graph. (SIGGRAPH Proc.)* (2004). 1, 2, 3

[MWM87] MASTEN G., WATTERBERG P., MAREDA I.: Fourier synthesis of ocean scenes. *IEEE Computer Graphics and Applications 7* (1987), 16–23. 2

[Ney03] NEYRET F.: Advected textures. In *Proceedings of Eurographics/SIGGRAPH Symposium on Computer Animation* (2003), pp. 147–153. 2

[OF02] OSHER S., FEDKIW R.: *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag, 2002. New York, NY. 3

[OH95] O'BRIEN J. F., HODGINS J. K.: Dynamic simulation of splashing fluids. In *Computer Animation '95* (Apr. 1995), pp. 198–205. 2

[Pan96] PANTON R. L.: *Incompressible Flow*. John Wiley & Sons, 1996. (2nd ed.). 5

[Pea86] PEACHEY D. R.: Modeling waves and surf. In *Computer Graphics (Proc. of SIGGRAPH 86)* (1986), vol. 20, pp. 65–74. 2

[Ped95] PEDERSEN H. K.: Decorating implicit surfaces. In *Proc. of SIGGRAPH 95* (1995), pp. 291–300. 2

[PTB*03] PREMOZE S., TASDIZEN T., BIGLER J., LEFOHN A., WHITAKER R.: Particle–based simulation of fluids. In *Comp. Graph. Forum (Eurographics Proc.)* (2003), vol. 22, pp. 401–410. 2

[Ree83] REEVES W. T.: Particle systems - a technique for modeling a class of fuzzy objects. In *Computer Graphics (Proc. of SIGGRAPH 83)* (1983), vol. 17, pp. 359–376. 2

[SF93] STAM J., FIUME E.: Turbulent Wind Fields for Gaseous Phenomena. In *Proc. of SIGGRAPH 1993* (1993), pp. 369–376. 2

[SHG*03] SUMNER N., HOON S., GEIGER W., MARINO S., RASMUSSEN N., FEDKIW R.: Melting a terminatrix. In *SIGGRAPH 2003 Sketches & Applications* (2003), ACM Press. 1

[Sta99] STAM J.: Stable fluids. In *Proc. of SIGGRAPH 99* (1999), pp. 121–128. 2, 3, 5

[Sta03] STAM J.: Flows on surfaces of arbitrary topology. *ACM Trans. Graph. (SIGGRAPH Proc.) 22*, 3 (2003), 724–731. 2

[Sus03] SUSSMAN M.: A second order coupled level set and volume-of-fluid method for computing growth and collapse of vapor bubbles. *J. Comp. Phys. 187* (2003), 110–136. 6

[Tes02] TESSENDORF J.: Simulating Ocean Water. In *SIGGRAPH 2002 Course Notes #9 (Simulating Nature: Realistic and Interactive Techniques)* (2002), ACM Press. 2

[TFK*03] TAKAHASHI T., FUJII H., KUNIMATSU A., HIWADA K., SAITO T., TANAKA K., UEKI H.: Realistic animation of fluid with splash and foam. *Comp. Graph. Forum (Eurographics Proc.) 22*, 3 (2003), 391–400. 2

[TMPS03] TREUILLE A., MCNAMARA A., POPOVIĆ Z., STAM J.: Keyframe control of smoke simulations. *ACM Trans. Graph. (SIGGRAPH Proc.) 22*, 3 (2003), 716–723. 2

[Wit99] WITTING P.: Computational fluid dynamics in a traditional animation environment. In *Proc. of SIGGRAPH 99* (1999), pp. 129–136. 2

[WR03] WRENNINGE M., ROBLE D.: Fluid simulation interaction techniques. In *SIGGRAPH 2003 Sketches & Applications* (2003), ACM Press. 2

© The Eurographics Association 2004.