Melting and Burning Solids into Liquids and Gases

Frank Losasso, Geoffrey Irving, Eran Guendelman, and Ron Fedkiw

***Abstract*— We propose a novel technique for melting and burning solid materials including the simulation of the resulting liquid and gas. The solid is simulated with traditional mesh-based techniques (triangles or tetrahedra) which enable robust handling of both deformable and rigid objects, collision and self-collision, rolling, friction, stacking, etc. The subsequently created liquid or gas is simulated with modern grid-based techniques including vorticity confinement and the particle level set method. The main advantage of our method is that state of the art techniques are used for both the solid and the fluid without compromising simulation quality when coupling them together or converting one into the other. For example, we avoid modeling solids as Eulerian grid-based fluids with high viscosity or viscoelasticity, which would preclude the handling of thin shells, self-collision, rolling, etc. Thus, our method allows one to achieve new effects while still using their favorite algorithms (and implementations) for simulating both solids and fluids, whereas other coupling algorithms require major algorithm and implementation overhauls and still fail to produce rich coupling effects (e.g. melting and burning solids).**

***Index Terms*— physically based modeling, melting, burning, solid, liquid, gas, phase change, Lagrangian mesh, Eulerian grid, adaptive mesh.**

I. Introduction

SIMULATIONS of water, smoke, and fire are becoming increasingly important in computer graphics applications such as feature films, since it is often costly, dangerous, or simply impossible to film the desired interactions between these fluids and their surroundings. These difficulties are exacerbated when the surrounding environment is undergoing complex motion or destructive modification, making it advantageous to combine fluid effects with simulations of rigid and deformable objects. Ideally, one would like to simultaneously incorporate the full range of available fluid and solid behaviors in a simulation including interactions between different components. This leads naturally to the concept of handling phase changes between solids and fluids.

Frank Losasso, Eran Guendelman, and Ron Fedkiw are with the Department of Computer Science, Stanford University, and Industrial Light and Magic. Geoffrey Irving is with the Department of Computer Science, Stanford University and Pixar Animation Studios. Email: {losasso,irving,erang,fedkiw}@cs.stanford.edu.

Historically, particle methods have been popular for the physical simulation of both solids and fluids as well as phase changes between them, and [1] applied them to softening and melting behavior in graphics early on. These methods avoid the need to maintain connectivity information for the solid phase, which simplifies topological change and transition between different material behaviors. Unfortunately, particle methods have not yet reached the quality and efficiency of specialized techniques for fluid or solid simulation. The lack of exact topological information for solids makes collision and self-collision problematic and thin sheets such as cloth almost impossible to simulate. High particle densities are typically required to achieve good accuracy for fluids, and it is often difficult to generate a smooth renderable surface from the resulting set of points. Moreover, while there has been recent progress in specialized methods for both solids (e.g. [2]) and fluids (e.g. [3]), researchers have not yet coupled together the specialized treatments devised for each. For example, [2] softens solids to the point where they start to melt and flow, but does not consider (or obtain) high quality particle based fluid effects as in [3].

The aim of our approach is to couple the highest quality simulation techniques for fluids and solids together, and thus we consider Eulerian grid-based techniques for fluids and Lagrangian mesh-based techniques for solids as in [4]. See also [5], although that method is not applicable to the deformable or thin materials treated in [4]. The opposite approaches of Lagrangian mesh-based fluids and Eulerian grid-based solids typically have only limited success, although there has been some interesting recent work with high viscosity and viscoelastic fluids. [6], [7] used high viscosity fluid to model melting and flowing, and [8] compute elastic forces resisting deformation by advecting a strain field along with the fluid. While these techniques allow the simulation of a much greater range of fluid phenomena, they do not support accurate self-collision, rolling, etc. as a tetrahedral mesh can, and cannot be applied to thin objects such as cloth.

For the fluid phase, we use state of the art simulation techniques for smoke [9], water [10] and fire [11]. We use standard Lagrangian mesh-based techniques for deformable objects (in particular [12]–[15]), except that we embed the surface geometry of the object in a parent

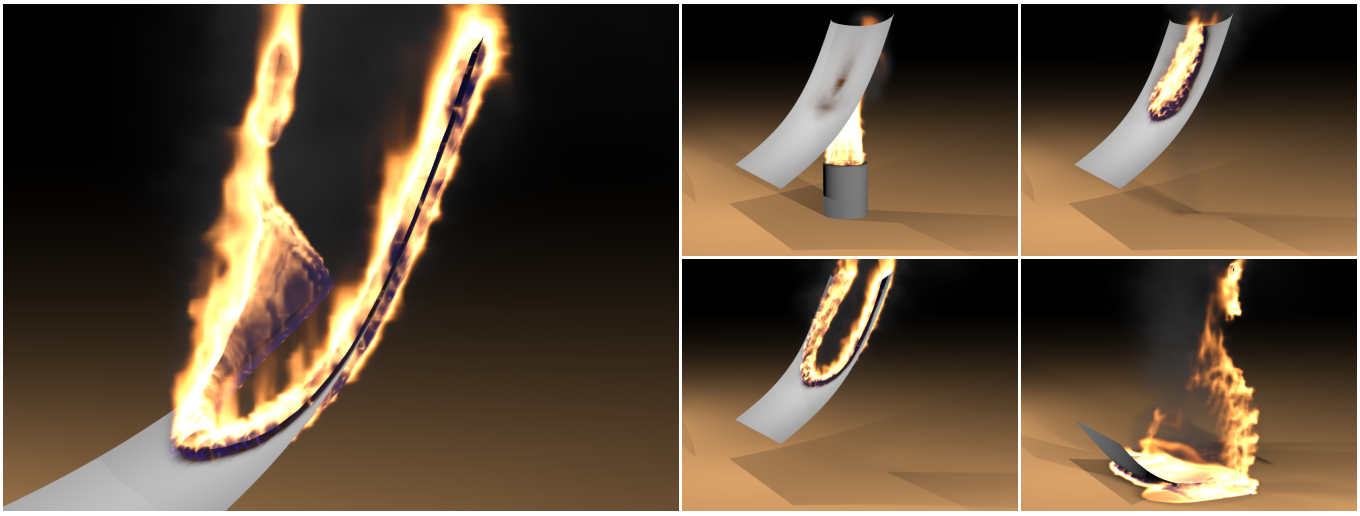

Fig. 1. A sheet of material constrained at its four corners catches fire. When the top corners burn away from the rest of the material, it falls to the ground (121x193x121 grid, 13K triangles).

simulation mesh as in [16] to allow smooth erosion of the surface during melting or burning. This is essentially a free form deformation (FFD) [17]. For more on FFD's applied to simulation, see for example [18]–[22].

[23], [24] used level sets to model the erosion of burning rigid objects (see also [25]), but did not consider thin objects, deformable objects or melting. We similarly use level set methods to simulate the erosion of solid material, but desire a direct implementation on the object's tetrahedral or triangular simulation mesh in material coordinates enabling the treatment of both thin and deformable objects. In order to avoid the difficulties associated with implementing level set algorithms on tetrahedral and triangular meshes, we instead evolve the level set on a background octree (or quadtree) grid in material coordinates interpolating the results to the simulation mesh. This allows us to directly leverage the level set algorithms proposed in [26] for octree data structures. The values of the level set function allow us to regenerate the evolving embedded geometry when necessary, and we use dynamic red-green refinement to increase the resolution of the simulation mesh near the boundary geometry as it changes over time. In fact, due to the special structure of our red-green refinement, the nodes of the simulation mesh correspond exactly to a subset of the nodes of the octree grid reducing interpolation to the trivial copying of data. Figure 1 shows examples of this technique applied to the burning of a material sheet.

II. Previous Work

[27] popularized the use of the incompressible three dimensional Navier-Stokes equations in computer graphics, and this work was subsequently extended with semi-Lagrangian advection and vorticity confinement in [28] and [9] respectively. Although it is not possible to discuss all the varied research in this direction, variations have been used to model fire [11], [29] and explosions [30], [31] (with compressible flow in [32]), control [33], [34], and flows on surfaces [35]. There is also the recent work of [36] that hybridizes grid-based methods with vortex particle techniques. Liquids have received a lot of focused attention including the early work of [37], [38], the particle and cell based approach of [39], [40], and the hybridized particle and level set approaches in [10], [41]. Additional work has been done on surface tension [26], [42], control [7], [43]–[45], solid fluid coupling [46], [47], contact angles [48], sand [49] and two phase flow [50].

III. Eroding Thin Shells

We simulate solids as triangular or tetrahedral meshes, and use standard element-based force computation and collision algorithms properly adjusted for embedding (see e.g. [16]). Since the solid is continuously eroding into fluid during melting or burning, we need to dynamically erode the mesh as material disappears while maintaining a smooth surface for collisions and rendering. Thus throughout the simulation, we adaptively refine the boundary elements of the simulation mesh to improve accuracy while preserving large elements elsewhere for efficiency. For a thin shell such as cloth where every element is on the surface requiring reasonable resolution, this adaptivity is primarily useful to resolve any complicated features at the cloth's edge (and was not required for our examples). However, for volumetric tetrahedral meshes, simulations are far more expensive if the invisible interior of the object is not discretized with larger elements. Since the algorithms and data structures required for two dimensional shells are quite similar to

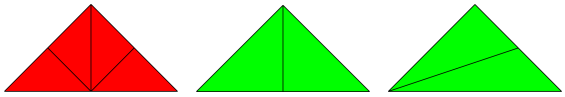

Fig. 2. Red and green refinement of BCS triangles.

those for three dimensional volumetric solids, we focus on the simpler case of thin shells first.

In order to support dynamic mesh adaptation, we begin with the uniform body-centered square (BCS) triangular lattice which is the two dimensional analog of the body-centered cubic (BCC) lattice used in mesh generation by [51], [52] (they also used red-green refinement similar to what we propose here). The BCS lattice is given by taking the vertices of a uniform square grid in space together with the centers of each square (the vertices of the dual grid), and placing four triangles in each square by connecting the four edges to the center node as shown in Figure 4 (upper left). We then perform red-green refinement initially labeling all BCS triangles as red. Any red triangle can then be red refined into four red triangles of half the size as shown in Figure 2. The resulting smaller triangles are exactly the BCS triangles from a grid of twice the resolution. After a sequence of red refinements, T-junctions are removed with green refinement which replaces a red triangle with two green triangles of possibly lower quality (Figure 2). Since we only allow one level of green refinement to ensure high quality elements, the initial red refinement is iterated until neighboring triangles are at most one level apart and no triangle has more than one T-junction. See Figure 4 (lower left) for an example of a red-green refined mesh. Note that if a green triangle later requires additional refinement, it and its sibling must be removed and replaced with a red refinement of the parent red triangle.

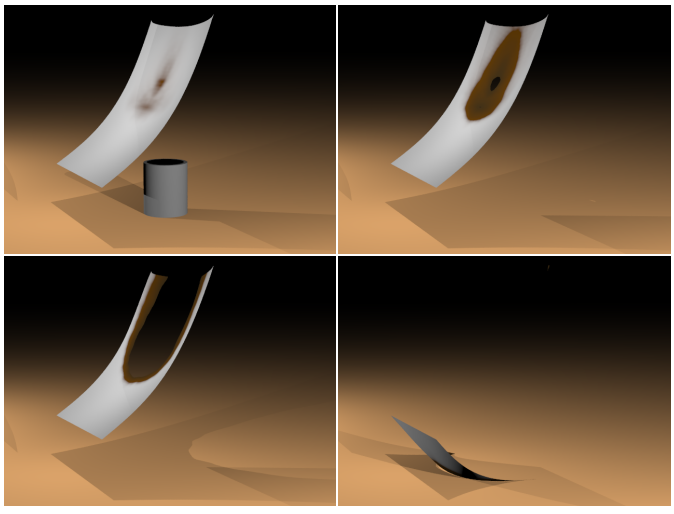

Fig. 3. A burning sheet of material rendered without the fire to illustrate the smooth boundary of the solid (13K triangles).

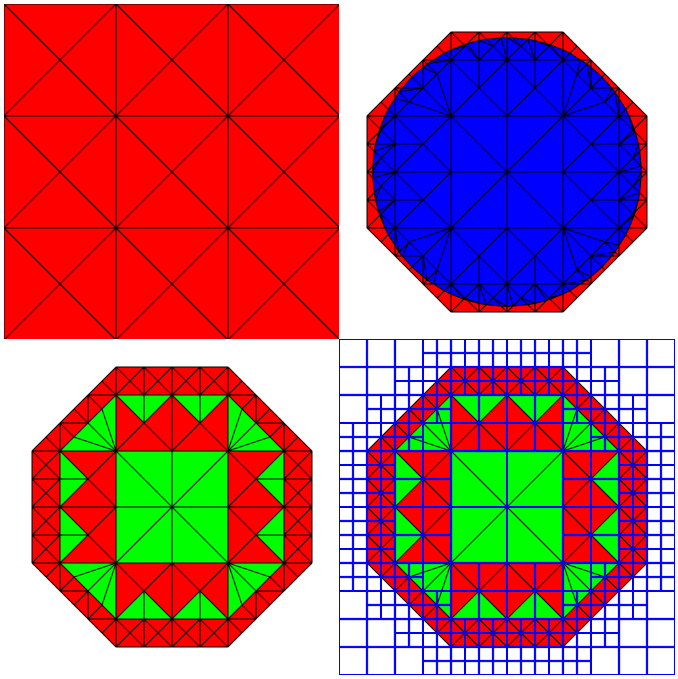

Fig. 4. Starting with a uniform body-centered square mesh in material space (upper left), we apply red-green refinement and discard triangles completely outside the object (upper right). The simulation is then performed on the red-green parent mesh (lower left). Level set operations are performed by overlaying a quadtree grid on top of the red-green structure, noting that each mesh vertex is also a node in the quadtree grid (lower right). A major advantage of this approach is that the front evolution is carried out on a quadtree mesh in the two-dimensional material space even as the object deforms in three spatial dimensions.

Since all triangles in a BCS mesh are aligned in a Cartesian grid structure, refinement alone is not enough to match a smooth boundary curve. [51], [52] solved this problem by compressing the mesh to the boundary before simulation. This is computationally infeasible in the context of melting or burning, since it would have to be performed after every step of the simulation. Instead, we leave the simulation mesh unchanged, and define the actual boundary of the object as the zero-isocontour of a level set function which is stored on the nodes of the simulation mesh. The surface of the object can be extracted directly from the level set function with marching triangles (the two-dimensional analog of marching tetrahedra), and any simulation triangles lying completely outside the level set can be removed. Figure 4 (upper right) shows the results obtained for a circular piece of geometry.

With the ability to spatially adapt our mesh based on the values of a level set function, simulating the loss of solid material (to the liquid or gas phase) is readily accomplished by smoothly increasing the nodal values of the level set function (and remeshing on the fly). However, calculating the change in level set values due to combustion, phase change, etc. can be

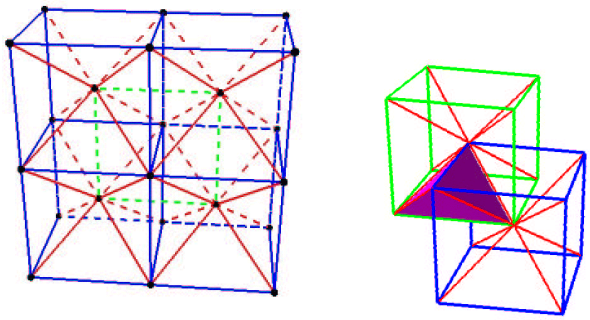

Fig. 5. A portion of the BCC lattice. The blue and the green connections depict the two interlaced grids, and the eight red connections at each node lace these two grids together.

daunting on our adaptive red-green data structure, since it typically requires the solution of partial differential equations on an only partially structured non-Cartesian BCS (or BCC) mesh. A key aspect of our approach is to avoid the solution of partial differential equations on arbitrary meshes with moving points, instead solving the governing equations for level set evolution in *material* space where the grid is static (i.e. grid points do not move). Moreover, we solve these equations on a structured Cartesian background grid interpolating data back and forth to the red-green simulation mesh. In fact, we choose a quadtree grid that *exactly* matches the red-green adaptive structure of the BCS mesh.

We emphasize that the match is exact, i.e., the vertices of the red-green simulation mesh lie exactly on top of the grid points of the quadtree grid as shown in Figure 4 (lower right). This means that there are no interpolation errors for moving data back and forth between the red-green mesh and the quadtree grid, and the computational cost is limited to the simple copying of data. Although *all* the red-green vertices correspond to quadtree nodes, there are some extra nodes in the quadtree grid that do not correspond to red-green vertices. Before computations can be carried out on the quadtree grid, values for these nodes must be interpolated or extrapolated from valid data on the red-green grid or the already defined quadtree grid nodes. If a quadtree node lies inside a red-green triangle, its value can be interpolated using barycentric coordinates. The remaining nodes can be filled in using the standard level set reinitialization and extrapolation equations on the quadtree grid. This background quadtree grid allows us to leverage all level set algorithms implemented on quadtree data structures including advection, motion by mean curvature, reinitialization, extrapolation of data across an interface or isocontour, the particle level set method, elliptic solvers, etc., see e.g. [26] and the references within. Note that if adaptivity is not needed, the use of an unrefined BCS mesh allows us to solve level set algorithms on an even simpler uniform grid.

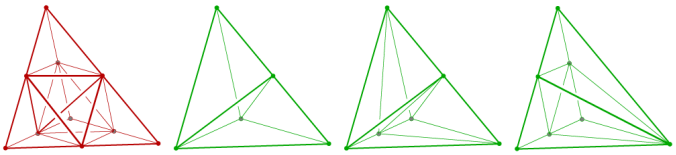

Fig. 6. Red refinement produces eight children that reside on a BCC lattice of half the size (left). Three types of green refinement are allowed in order to remove T-junctions (right).

While the red-green refinement strategy applies to any mesh, the quadtree correspondence applies only to grid aligned meshes. In particular, it does not hold for equilateral triangle meshes. Also, while the back and forth interpolation strategy used for updating the level set function could be implemented with a different background grid, the matching quadtree grid gives an optimal match for vertices and refinement structure minimizing computational cost and interpolation error.

Once the red-green mesh and boundary are constructed, the simulation proceeds along standard lines. Values such as world position and velocity are interpolated to new nodes as they are generated during refinement. Values from the simulation mesh are also interpolated to the embedded boundary, which is used for collision, self-collision and fluid interaction.

IV. Eroding Volumetric Solids

Two-dimensional geometric methods do not typically carry over elegantly to three spatial dimensions. However, we designed our two-dimensional approach with three dimensions in mind, and thus the method carries over almost entirely. In three spatial dimensions, we tile space with a body-centered cubic (BCC) lattice and form tetrahedral elements by considering the vertices of the uniform cubic grid together with the cell centers.

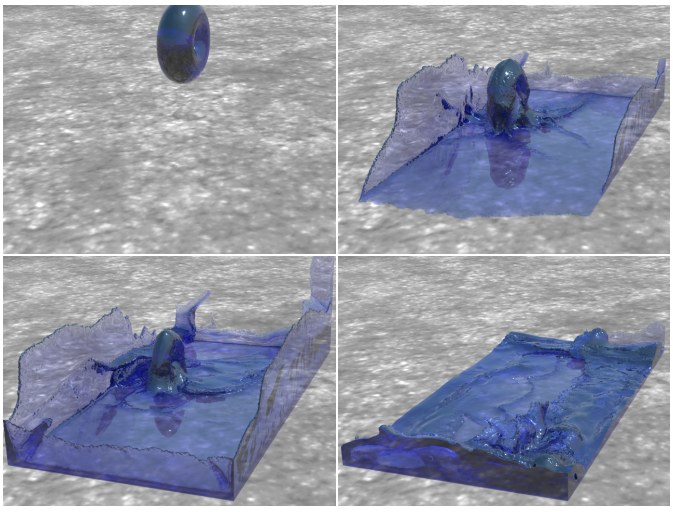

Fig. 7. An elastic torus bounces and rolls on the ground as it melts, generating thin sheets of liquid (242x121x121 grid, 22K tetrahedra).

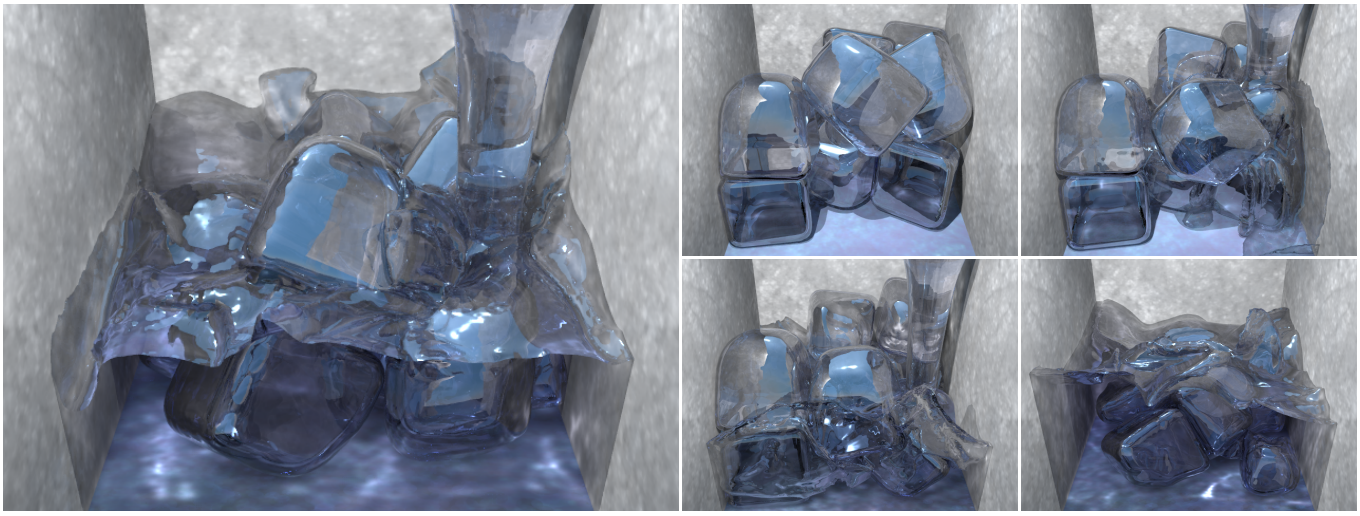

Fig. 8. Ten rigid body simulated ice cubes are melted by a stream of hot water (100^3 grid, 600K total surface triangles).

Space is divided into octahedrons by considering the region connecting each cubic face to its neighboring cell centers, and four tetrahedra are placed in each octahedron. See Figure 5. Red refinement of a tetrahedron produces eight smaller BCC tetrahedra, and T-junctions are removed using several varieties of green refinement which result in two or four green children. See Figure 6. Overall, our approach is similar to that in [51], [52], and we refer the reader to their work for further details.

Once again, we define a level set function on the nodes of the red-green simulation mesh, and use marching tetrahedra and embedded simulation technology similar to that proposed in [16]. Most importantly, we implement all our level set methods on an overlayed background octree grid in material space. Just as in two spatial dimensions, the nodes of the octree grid correspond exactly to those of the tetrahedra in the adaptive red-green simulation mesh. This makes the method both computationally efficient and straightforward to implement leveraging on [26] and the references therein.

For ease of implementation, rigid body surfaces are generated in the same manner as the deformable case, even though the tetrahedral mesh is not used during rigid body dynamics. This incurs no significant computational cost, since the vast majority of the total simulation time is spent solving for the fluid dynamics.

V. Creating Fluids from Solids

In the case of melting, the solid and liquid phase have similar densities, so a given volume of solid material is converted into approximately the same volume of liquid. In contrast, the density ratio between a solid and its gaseous products is typically several orders of magnitude, and a large volume of gas is released before any noticeable amount of solid disappears. Thus, we simulate the production of fluid from solid in very different ways for the melting and burning cases. Moreover, we do not consider burning volumetric solids, since it takes a long time to see changes in the solid. Instead, we consider burning shells where there is very little solid material, and thus it erodes away quickly. In addition, we only consider the melting of volumetric objects, since melting shells produce only a small amount of liquid that is difficult to resolve on a computational fluid dynamics grid.

VI. Creating Liquid

We model the liquid phase using the particle level set method of [10]. The bulk of the fluid is represented by a level set function defined on the grid with negative values denoting liquid and positive values denoting air. The level set provides clean handling of topological merging and breaking, but suffers from volume loss in regions of high curvature due to numerical diffusion.

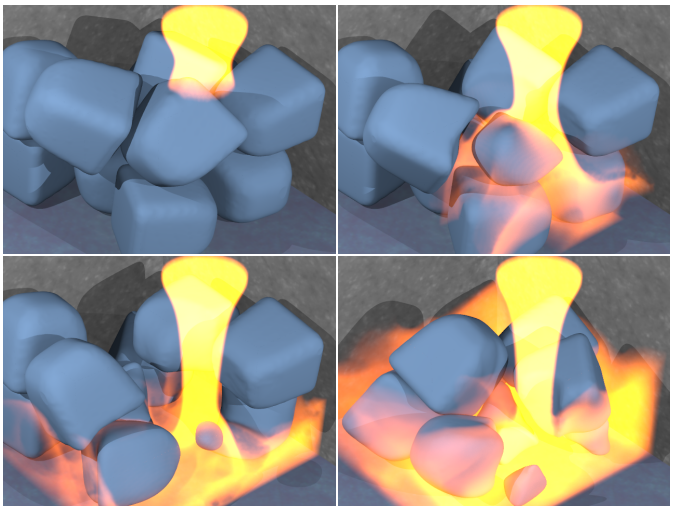

Fig. 9. The melting speed of the ice cubes is derived from a temperature field advected with the liquid velocity.

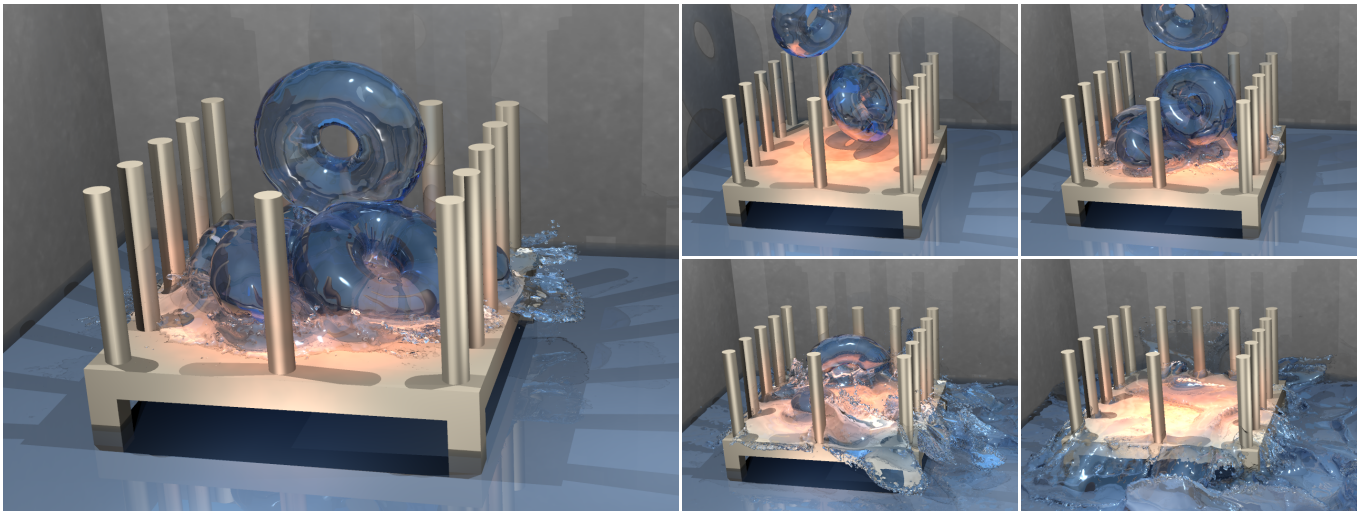

Fig. 10. Six elastic tori fall onto a hot surface causing them to quickly melt into liquid (181x61x181 grid, 160K total tetrahedra).

To alleviate this problem, [41] proposed the use of particles passively advected with the fluid velocity to replace liquid lost to dissipation. After each advection step, each particle was treated as a small sphere and added to the level set with a CSG union. [10] later extended this technique by adding a layer of positive particles outside the fluid in addition to the negative particles inside, which reduces the opposite problem of volume gain (air loss) due to dissipation in the other direction. That and other proposed modifications made the method accurate enough for scientific calculations as well as visually compelling. Since particles exist only in a thin band around the interface and do not need to interact, they can be seeded at higher resolution than the fluid grid to improve accuracy. The particle radii used for level set modification are therefore much smaller than a grid cell, which may result in particles failing to convert a surrounding node to liquid and incorrectly crossing over the the level set zero isocontour. These *removed* particles move ballistically and collide with objects until they occur in high enough density to convert grid nodes into fluid. [41] rendered these removed particles as spray droplets.

As the solid melts, we generate liquid at the surface of the solid using techniques synergistic with the particle level set method. At the beginning of the simulation, we seed liquid particles inside the solid in material space using the same particle density and radius distribution used for the particle level set method when modeling the liquid. Note that this seeding is readily implemented on the quadtree or octree grid in material space. Then, as the level set evolves on the material space quadtree or octree grid, we make note of any particles that cross completely over the level set based on their radii effectively moving outside the solid. These particles are moved from material space to world space (in the same way embedded particles are carried along by the simulation mesh) and used to generate fluid. Fluid is generated by adding these particles directly to the particle level set simulation as removed particles with an initial velocity determined as if the particle were an embedded particle. Some of these particles will be in free flight and generate new fluid (by modifying the level set function in the usual manner), while others will already be interior to the fluid and thus immediately reincorporated into the particle level set simulation as negative particles. This allows thin sheets of melted liquid to form on the object surface before it would be possible to resolve the flow on the fluid dynamics grid. This process is illustrated in Figure 11.

The negative particles in the particle level set method are used only to resolve the liquid interface, and do not otherwise influence the physics of the liquid. Thus they have no mass or volume information, and do not interact with each other except through the Eulerian grid. If these particles venture too far across the interface into the air region and do not occur in dense enough populations to create new liquid volume, they become removed particles and are treated ballistically as droplets in free flight. This

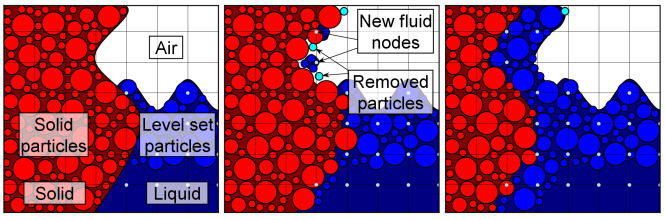

Fig. 11. A melting solid partially immersed in liquid. Particles are seeded throughout the solid with the same distribution and properties as the particle level set particles (left). As the solid melts, particles leave the solid and are converted to removed liquid particles. Some of these particles are close enough to grid nodes to convert them to liquid (middle). More liquid is generated as the solid continues melting (right).

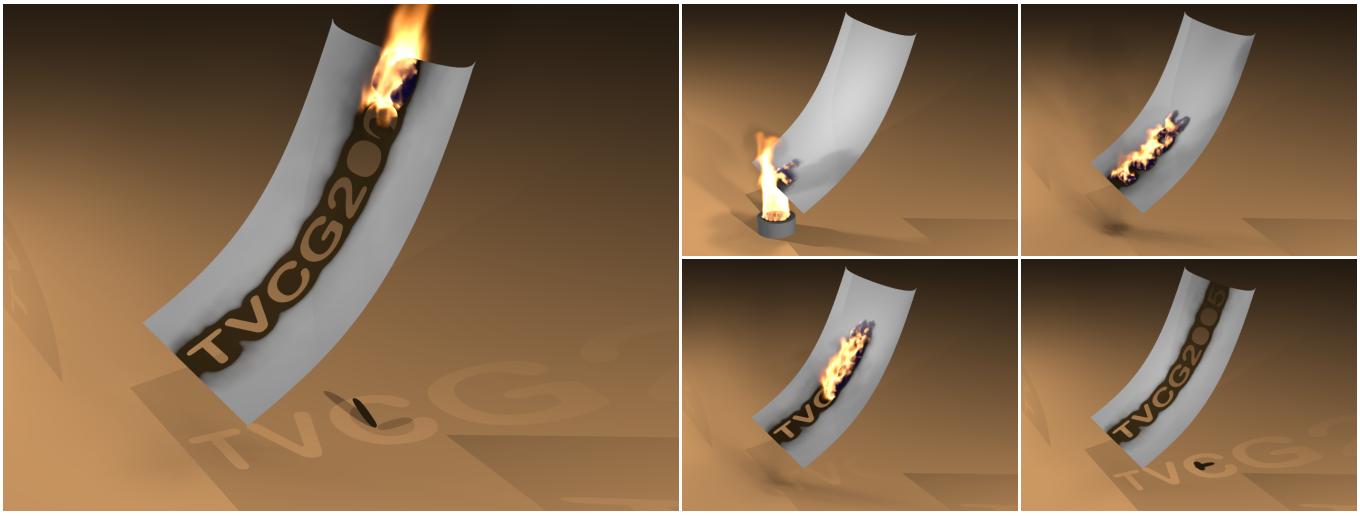

Fig. 12. A predefined level set is used to prescore part of a thin material sheet as inflammable (121x193x85 grid, 37K triangles).

could likely be improved by modeling these removed particles with a smoothed particle hydrodynamics (SPH) method such as that proposed in [3], [53]. Additionally, the negative particles interior to the liquid volume could be used as boundary conditions for the removed particle SPH simulation. Regardless, even without this, our approach produces compelling animations.

VII. Creating Gas

We use the fire simulation method of [11], which models fire as a blue core of unburnt fuel undergoing a combustion reaction into hot gaseous products. A level set is used to track the location of the blue core, and the products are described by temperature and density fields. Since the density of the gaseous fuel is higher than the resulting products, the gas expands as it crosses the level set interface and is converted from fuel to products. The rate of this expansion can be derived by solving the jump conditions for conservation of mass and momentum as described in [11]. To model the sourcing of fuel from kinematic solid objects, they constrain the velocities at the boundary of the object to inject fuel at a given rate (which can also be determined from the jump conditions).

When a region of solid catches fire and begins burning, we continuously source gaseous fuel around the burning region while maintaining a reaction coordinate on the vertices of the solid describing the progress of the combustion reaction. This can also be used for browning effects at render time. We model the expansion from solid into gaseous fuel by generating a thin band of fuel level set around the burning region and setting the divergence in this region to a positive value when solving for the pressure forcing the fuel to move outwards. The amount of gas produced can be directly linked to the rate at which the solid is disappearing, although the animator can of course control this effect as desired. Divergence sourcing was also used in [30] to model suspended particle explosions by making each suspended dust particle a small divergence source.

VIII. Two Way Solid Fluid Coupling

We follow the solid fluid coupling strategy proposed in [4] which allows one to use their favorite (and previously implemented) methods for simulating the solid and the fluid independent of the coupling strategy, unlike [5] which requires special treatment of the solid, i.e. treating it as some sort of fluid in order to obtain two way coupling. Thus while [5] can only handle rigid volumetric bodies that can be rasterized and represented on the fluid grid, [4] showed full two way coupling for deformable bodies as well as very thin objects such as cloth. The main philosophical difference between the two approaches is that [4] computes the effect of the fluid on the solid via a force per vertex that can be added to *any* solid simulation technique, while [5] allows the fluid to completely determine the solid's velocity overwriting all internal dynamics (such as elastic deformation). In addition, the final projection step of [5] takes the continuous velocity field computed for the combined solid/fluid domain and constrains the grid cells within the solid to have a velocity consistent with rigid body motion. This makes the velocity field discontinuous at the surface of the solid, allowing fluid to flow directly into the solid and disappear. In contrast, [4] applies Neumann boundary conditions at the surface of the solid enforcing the fluid's normal velocity to be identical to that of the solid at the solid/fluid interface.

[4] focused on thin objects, whereas we are also interested in volumetric objects. Thus, we review some of their techniques as well as propose a couple of algorithmic improvements. Recall that the projection method

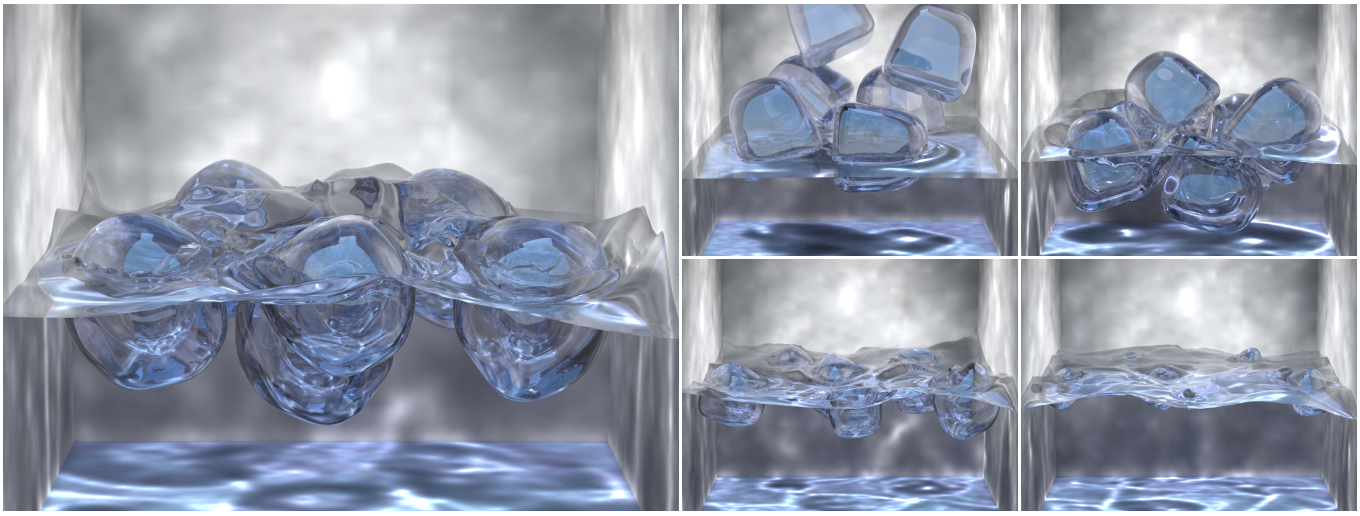

Fig. 13. Rigid ice cubes floating and melting in water with full two-way force coupling (100^3 grid, 600K total surface triangles).

for incompressible flow first computes an intermediate velocity $\mathbf{u}^*$ and then solves for the pressure p needed to make the final velocity $\mathbf{u}^{n+1}$ divergence free:

$$\mathbf{u}^* = \mathbf{u}^n - \Delta t(\mathbf{u}^n \cdot \nabla)\mathbf{u}^n + \Delta t\mathbf{g} \quad (1)$$

$$\nabla \cdot (\nabla p/\rho) = \nabla \cdot \mathbf{u}^*/\Delta t \quad (2)$$

$$\mathbf{u}^{n+1} = \mathbf{u}^* - \Delta t \nabla p/\rho \quad (3)$$

We use the new node based fluid solver of [4] which stores the permanent velocity on the cell faces of a MAC grid. At the beginning of each time step, the face velocities are averaged to the nodes and used in Equation 1 to compute $\mathbf{u}^*$ at the nodes. Then we calculate a scaled force $\Delta\mathbf{u} = \mathbf{u}^* - \mathbf{u}^n$ at each node, average this back to the faces, and use the scaled force at each face to increment the face velocity to its intermediate value. Finally, the standard MAC grid based pressure solver is used to make this intermediate face velocity divergence free. This scheme allows one to calculate the intermediate velocity using a simple node based method, while still using a standard MAC grid approach to solve for the pressure. Moreover, it avoids direct averaging of the velocity field significantly reducing numerical dissipation that gives the fluid an unwanted viscous appearance. When solving for the pressure, Neumann boundary conditions are applied at cell faces located within solid objects using the *effective velocity* of the solid, which is calculated by evolving the solid forward in time by the size of the *next* time step. Using the effective velocity (as opposed to the instantaneous velocity) greatly reduces mass loss, especially when thin liquid films interact with solid objects.

In [4], they first compute the intermediate fluid velocity $\mathbf{u}^*$ using knowledge of the solid's position at both time n and time $n+1$. Then they solve the variable density Poisson equation $\nabla \cdot (\nabla p/\rho) = \nabla \cdot \mathbf{u}^{old}/\Delta t$ for the solid fluid coupling pressure. The density at each face is set to either the fluid's density or the solid's density, and $\mathbf{u}^{old}$ is set to $\mathbf{u}^*$ in the fluid region and to the effective solid velocity inside the object. The resulting pressure is interpolated to the barycenter of each triangle and multiplied by the triangle area and normal to obtain the net force on a face. For rigid bodies the force and torque are calculated by accounting for all the triangles, and for deformable objects one third of the net force is distributed to each node. These forces are used when the solid is evolved from time $n+1$ to time $n+2$, and that motion is used to compute the *effective velocity* which is used to project the intermediate fluid velocity $\mathbf{u}^*$ to be divergence free at time $n+1$ concluding the time step from n to $n+1$.

The pressures are defined at the cell centers of a MAC grid making interpolation to the barycenter of triangles difficult near the boundaries of the domain. This can be remedied by extrapolating pressure values into a one ring of ghost cells surrounding the domain, but this causes errors in the case of solid walls. We propose a new method that accurately calculates pressure derivatives at solid wall cell faces on the boundary of the domain, and then we use those derivatives to fill in pressure values in the ghost cells assuming a linear pressure profile across each cell face. (Of course, this same method can be used for kinematic interior boundaries as well.) To do this we point out that the pressure calculated from the variable density Poisson equation is typically used to update the fluid velocity via equation 3, although we do not do this since this pressure is merely used to calculate the force the fluid applies to the solid. We instead use the appropriate component of this equation to solve for the pressure derivative at a wall using the density of the fluid for ρ and the final known wall velocity u^{new} for the new velocity. For example, the

Fig. 14. Burning cloth draped over an armadillo figurine (89x122x78 grid, 18K triangles).

pressure derivative across a face normal to the x-axis would be $p_x = \rho(u^{old} - u^{new})/\Delta t$. The old velocity is calculated by updating the nodes on the wall to their $\mathbf{u}^*$ values and averaging these values to the cell face. Note that substituting this pressure derivative into the variable density Poisson equation leads to the replacement of the *old* velocity value with the *new* value, which is *exactly* equivalent to the standard practice of setting the pressure derivative to zero and using the new solid velocity value on a face. Thus one does not need to calculate $\mathbf{u}^*$ on the cell face unless the *true* (nonzero) pressure derivative is desired there.

Besides this improved treatment of the pressure near kinematic boundaries, we also propose improving the overall accuracy of the fluid to solid coupling forces. The main idea is to compute a more accurate representation for $\mathbf{u}^{old}$ in the cells occupied by the object. For example, consider a stationary neutrally buoyant object submerged in a still fluid, where it should remain at rest. In order for the variable density Poisson equation $\nabla \cdot (\nabla p/\rho) = \nabla \cdot \mathbf{u}^{old}/\Delta t$ to give the correct coupling pressure, $\mathbf{u}^{old}$ inside the solid should be identical to the fluid's $\mathbf{u}^*$, which is $\Delta t \mathbf{g}$. If $\mathbf{u}^{old} = \mathbf{0}$ was used inside the solid region (corresponding to the solid's instantaneous or effective velocity), an incorrect coupling pressure would arise due to the incompatibility between the solid and fluid velocity causing the solid to incorrectly accelerate. Instead of using the effective velocity of the solid which represents the change in the solids position from time n to time $n+1$, we subtract out the effects of the fluid to solid pressure coupling that was used to update the solid from time n to time $n+1$. This velocity is more accurately in line with the intermediate $\mathbf{u}^*$ velocity computed for the fluid, which accounts for all the forces except those due to the pressure.

IX. Examples

For all our examples, the additional cost of melting or burning was small compared to the time required to couple solid and fluid together without phase change. The total computational cost averaged between 5 and 15 minutes per frame.

The single torus in Figure 7 was melted at a constant rate by uniformly increasing the values of the level set function on the octree grid in material space. For the other melting examples, we set the melting velocity to be a linear function of temperature multiplied by the level set normal on the octree background grid. In the multiple tori example (Figure 10), the temperature was defined procedurally to be large in a band near the table. In Figure 8, we inject hot water advecting the temperature with the liquid volume (Figure 9). Figure 13 shows ice cubes melting after being dropped into warm water. For rendering, we place a small ellipsoid around both negative and removed negative particles and blend it with the level set before ray tracing. The aspect ratios of the ellipses are based on the velocity of the associated particle.

To model the material catching on fire in Figures 1, 12 and 14 we store a temperature field on the nodes of the simulation triangle mesh and gradually relax that temperature towards the temperature of the surrounding air, igniting the material when a prescribed ignition temperature is reached. Then we use a bandwidth of 2 grid cells to generate a fuel level set around the solid, and apply divergence sourcing in a band of 2.5 grid cells. The solid level set function is set to $\phi = Y - Y_{max}$, where Y is the reaction coordinate and Y_{max} indicates full combustion, causing the surface to erode once the reaction completes.

Since we have complete control over the level set

function defining the boundary of the solid, we can readily control the pattern of melting or burning. In Figure 12, a region of the material is constrained to be inflammable by performing a CSG union with a fixed shape and sourcing fuel only from the flammable region.

X. Conclusions and Future Work

We have presented a novel technique for simulating the phase change of solid objects modeled by Lagrangian meshes into fluids defined on Eulerian grids. Examples were presented to demonstrate that this algorithm works well for both the melting of volumetric solids into liquid and the burning of thin sheets of material into gas. Simulation meshes for the solid are constructed by adaptive red-green refinement starting with a BCS or BCC lattice, and the boundary of the object is specified by a level set function defined on the nodes of the simulation mesh which is evolved on a background quadtree or octree grid in material space. The most important benefit of the method is the ability to use state of the art techniques for both the solid and fluid without compromising simulation quality in order to couple them together or convert one into the other.

There are several areas for improvement for both the solid and fluid phases. For solids, significant work is needed to make embedded collision handling as accurate as the nonembedded case. Moreover, while embedded meshes allow for smooth changes in geometry, the mass distribution changes discontinuously when simulation nodes disappear, which can result in small popping artifacts. This could be alleviated by adjusting the masses of boundary points as smoothly as possible while clamping them away from zero for stability. Finally, the use of the BCS-quadtree correspondence for arbitrarily curved shells would require dividing the surface up into rectangular patches. Fortunately, the quadtree correspondence is unnecessary in the burning case (at least for our ad-hoc model) since the level set is defined procedurally and all other steps in the algorithm carry over to an irregular mesh (including red-green refinement). In the case of melting volumetric objects, the octree correspondence is used frequently but does not restrict the class of objects which can be simulated.

Both melting and burning require the representation of details at close to the resolution of the grid, and some aliasing is visible in the burning simulations. Some of this is due to the use of linear interpolation during rendering, and could be dramatically improved with smoother thin shell-aware interpolation schemes. Further improvement would require better object boundary conditions for the fluid solver.

XI. Acknowledgements

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, ARO DAAD19-03-1-0331, NSF ITR-0121288, NSF ACI-0323866, NSF IIS-0326388 and NSF ITR-0205671. G.I. was supported in part by a National Science Foundation Graduate Research Fellowship. We'd like to thank Tamar Shinar and Andrew Selle for help with rendering, Mike Houston, Christos Kozyrakis, Mark Horowitz, Bill Dally and Vijay Pande for computing resources at Stanford, and Chris Walker for computing resources at Pixar Animation Studios.

References

[1] D. Terzopoulos, J. Platt, and K. Fleischer, "Heating and melting deformable models (from goop to glop)," in *Graphics Interface*, 1989, pp. 219–226.

[2] M. Müller, R. Keiser, A. Nealen, M. Pauly, M. Gross, and M. Alexa, "Point based animation of elastic, plastic and melting objects," in *Proc. of the 2004 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, 2004, pp. 141–151.

[3] S. Premoze, T. Tasdizen, J. Bigler, A. Lefohn, and R. Whitaker, "Particle–based simulation of fluids," in *Comp. Graph. Forum (Eurographics Proc.)*, vol. 22, no. 3, 2003, pp. 401–410.

[4] E. Guendelman, A. Selle, F. Losasso, and R. Fedkiw, "Coupling water and smoke to thin deformable and rigid shells," *ACM Trans. Graph. (SIGGRAPH Proc.)*, vol. 24, no. 3, pp. 973–981, 2005.

[5] M. Carlson, P. J. Mucha, and G. Turk, "Rigid fluid: Animating the interplay between rigid bodies and fluid," *ACM Trans. Graph. (SIGGRAPH Proc.)*, vol. 23, pp. 377–384, 2004.

[6] M. Carlson, P. Mucha, R. Van Horn, and G. Turk, "Melting and flowing," in *ACM SIGGRAPH Symposium on Computer Animation*, 2002, pp. 167–174.

[7] N. Rasmussen, D. Enright, D. Nguyen, S. Marino, N. Sumner, W. Geiger, S. Hoon, and R. Fedkiw, "Directible photorealistic liquids," in *Proc. of the 2004 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, 2004, pp. 193–202.

[8] T. G. Goktekin, A. W. Bargteil, and J. F. O'Brien, "A method for animating viscoelastic fluids," *ACM Trans. Graph. (SIGGRAPH Proc.)*, vol. 23, pp. 463–467, 2004.

[9] R. Fedkiw, J. Stam, and H. Jensen, "Visual simulation of smoke," in *Proc. of ACM SIGGRAPH 2001*, 2001, pp. 15–22.

[10] D. Enright, S. Marschner, and R. Fedkiw, "Animation and rendering of complex water surfaces," *ACM Trans. Graph. (SIGGRAPH Proc.)*, vol. 21, no. 3, pp. 736–744, 2002.

[11] D. Nguyen, R. Fedkiw, and H. Jensen, "Physically based modeling and animation of fire," in *ACM Trans. Graph. (SIGGRAPH Proc.)*, vol. 29, 2002, pp. 721–728.

[12] R. Bridson, R. Fedkiw, and J. Anderson, "Robust treatment of collisions, contact and friction for cloth animation," *ACM Trans. Graph. (SIGGRAPH Proc.)*, vol. 21, pp. 594–603, 2002.

[13] R. Bridson, S. Marino, and R. Fedkiw, "Simulation of clothing with folds and wrinkles," in *Proc. of the 2003 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, 2003, pp. 28–36.

[14] E. Guendelman, R. Bridson, and R. Fedkiw, "Nonconvex rigid bodies with stacking," *ACM Trans. Graph. (SIGGRAPH Proc.)*, vol. 22, no. 3, pp. 871–878, 2003.

[15] G. Irving, J. Teran, and R. Fedkiw, "Invertible finite elements for robust simulation of large deformation," in *Proc. of the ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, 2004, pp. 131–140.

[16] N. Molino, Z. Bao, and R. Fedkiw, "A virtual node algorithm for changing mesh topology during simulation," *ACM Trans. Graph. (SIGGRAPH Proc.)*, vol. 23, pp. 385–392, 2004.

[17] T. Sederberg and S. Parry, "Free-form deformations of solid geometric models," *Comput. Graph. (SIGGRAPH Proc.)*, pp. 151–160, 1986.

[18] P. Faloutsos, M. van de Panne, and D. Terzopoulos, "Dynamic free-form deformations for animation synthesis," *IEEE Trans. on Vis. and Comput. Graphics*, vol. 3, no. 3, pp. 201–214, 1997.

[19] S. Capell, S. Green, B. Curless, T. Duchamp, and Z. Popović, "A multiresolution framework for dynamic deformations," in *ACM SIGGRAPH Symp. on Comput. Anim.* ACM Press, 2002, pp. 41–48.

[20] S. Capell, S. Green, B. Curless, T. Duchamp, and Z. Popović, "Interactive skeleton-driven dynamic deformations," *ACM Trans. Graph. (SIGGRAPH Proc.)*, vol. 21, pp. 586–593, 2002.

[21] M. Müller, M. Teschner, and M. Gross, "Physically-based simulation of objects represented by surface meshes," in *Proc. Comput. Graph. Int.*, June 2004, pp. 156–165.

[22] D. James, J. Barbic, and C. Twigg, "Squashing cubes: Automating deformable model construction for graphics," in *SIGGRAPH 2004 Sketches & Applications*. ACM Press, 2004.

[23] Z. Melek and J. Keyser, "Interactive simulation of burning objects," in *Pacific Graph.*, 2003, pp. 462–466.

[24] Z. Melek and J. Keyser, "Multi-representation interaction for physically based modeling," in *ACM Symp. on Solid and Physical Modeling*, 2005, pp. 187–196.

[25] Y. Zhao, X. Wei, Z. Fan, A. Kaufman, and H. Qin, "Voxels on fire," in *Proceedings of IEEE Visualization*, 2003, pp. 271–278.

[26] F. Losasso, F. Gibou, and R. Fedkiw, "Simulating water and smoke with an octree data structure," *ACM Trans. Graph. (SIGGRAPH Proc.)*, vol. 23, pp. 457–462, 2004.

[27] N. Foster and D. Metaxas, "Modeling the motion of a hot, turbulent gas," in *Proc. of SIGGRAPH 97*, 1997, pp. 181–188.

[28] J. Stam, "Stable fluids," in *Proc. of SIGGRAPH 99*, 1999, pp. 121–128.

[29] A. Lamorlette and N. Foster, "Structural modeling of natural flames," *ACM Trans. Graph. (SIGGRAPH Proc.)*, vol. 21, no. 3, pp. 729–735, 2002.

[30] B. E. Feldman, J. F. O'Brien, and O. Arikan, "Animating suspended particle explosions," *ACM Trans. Graph. (SIGGRAPH Proc.)*, vol. 22, no. 3, pp. 708–715, 2003.

[31] N. Rasmussen, D. Nguyen, W. Geiger, and R. Fedkiw, "Smoke simulation for large scale phenomena," *ACM Trans. Graph. (SIGGRAPH Proc.)*, vol. 22, pp. 703–707, 2003.

[32] G. Yngve, J. O'Brien, and J. Hodgins, "Animating explosions," in *Proc. SIGGRAPH 2000*, vol. 19, 2000, pp. 29–36.

[33] A. Treuille, A. McNamara, Z. Popović, and J. Stam, "Keyframe control of smoke simulations," *ACM Trans. Graph. (SIGGRAPH Proc.)*, vol. 22, no. 3, pp. 716–723, 2003.

[34] R. Fattal and D. Lischinski, "Target-driven smoke animation," *ACM Trans. Graph. (SIGGRAPH Proc.)*, vol. 23, pp. 441–448, 2004.

[35] J. Stam, "Flows on surfaces of arbitrary topology," *ACM Trans. Graph. (SIGGRAPH Proc.)*, vol. 22, pp. 724–731, 2003.

[36] A. Selle, N. Rasmussen, and R. Fedkiw, "A vortex particle method for smoke, water and explosions," *ACM Trans. Graph. (SIGGRAPH Proc.)*, vol. 24, no. 3, pp. 910–914, 2005.

[37] M. Kass and G. Miller, "Rapid, stable fluid dynamics for computer graphics," in *Computer Graphics (Proc. of SIGGRAPH 90)*, vol. 24, no. 4, 1990, pp. 49–57.

[38] J. Chen and N. Lobo, "Toward interactive-rate simulation of fluids with moving obstacles using the navier-stokes equations," *Computer Graphics and Image Processing*, vol. 57, pp. 107–116, 1994.

[39] N. Foster and D. Metaxas, "Realistic animation of liquids," *Graph. Models and Image Processing*, vol. 58, pp. 471–483, 1996.

[40] N. Foster and D. Metaxas, "Controlling fluid animation," in *Computer Graphics International 1997*, 1997, pp. 178–188.

[41] N. Foster and R. Fedkiw, "Practical animation of liquids," in *Proc. of ACM SIGGRAPH 2001*, 2001, pp. 23–30.

[42] J.-M. Hong and C.-H. Kim, "Animation of bubbles in liquid," *Comp. Graph. Forum (Eurographics Proc.)*, vol. 22, no. 3, pp. 253–262, 2003.

[43] A. McNamara, A. Treuille, Z. Popović, and J. Stam, "Fluid control using the adjoint method," *ACM Trans. Graph. (SIGGRAPH Proc.)*, pp. 449–456, 2004.

[44] V. Mihalef, D. Metaxas, and M. Sussman, "Animation and control of breaking waves," in *Proc. of the 2004 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, 2004, pp. 315–324.

[45] L. Shi and Y. Yu, "Taming liquids for rapidly changing targets," in *Proc. of the ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, 2005, pp. 229–236.

[46] O. Génevaux, A. Habibi, and J.-M. Dischler, "Simulating fluid-solid interaction," in *Graphics Interface*, June 2003, pp. 31–38.

[47] M. Müller, S. Schirm, M. Teschner, B. Heidelberger, and M. Gross, "Interaction of fluids with deformable solids," *J. Comput. Anim. and Virt. Worlds*, vol. 15, no. 3–4, pp. 159–171, July 2004.

[48] H. Wang, P. Mucha, and G. Turk, "Water drops on surfaces," *ACM Trans. Graph. (SIGGRAPH Proc.)*, vol. 24, no. 3, pp. 921–929, 2005.

[49] Y. Zhu and R. Bridson, "Animating sand as a fluid," *ACM Trans. Graph. (SIGGRAPH Proc.)*, vol. 24, no. 3, pp. 965–971, 2005.

[50] J.-M. Hong and C.-H. Kim, "Discontinuous fluids," *ACM Trans. Graph. (SIGGRAPH Proc.)*, vol. 24, no. 3, pp. 915–919, 2005.

[51] N. Molino, R. Bridson, J. Teran, and R. Fedkiw, "A crystalline, red green strategy for meshing highly deformable objects with tetrahedra," in *12th Int. Meshing Roundtable*, 2003, pp. 103–114.

[52] R. Bridson, J. Teran, N. Molino, and R. Fedkiw, "Adaptive physics based tetrahedral mesh generation using level sets," *Eng. w. Comp.*, 2005.

[53] M. Müller, D. Charypar, and M. Gross, "Particle-based fluid simulation for interactive applications," in *Proc. of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2003, pp. 154–159.