Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2007)
D. Metaxas and J. Popovic (Editors)

A Simple Boiling Module

Theodore Kim[1] and Mark Carlson[2]

[1]IBM TJ Watson Research Center
[2]Walt Disney Animation Studios

Abstract

Recent efforts to visually capture the phenomena of boiling have proposed monolithic approaches that extend the basic techniques underlying existing fluid solvers. In this work, we show that if we instead treat boiling as a separate computational module to be loosely coupled to an existing solver, a very easy to implement, highly efficient algorithm can be designed that produces excellent visual results, even on coarse (64^3) grids. The algorithm is also highly SIMD-amenable, allowing the boiling computation to be farmed out to a GPU or Playstation 3 Cell processor. Our algorithm takes less than 100 lines of commented, readable C++, and can be integrated into an existing particle level set fluid solver with virtually no modifications. A serial implementation consumes between 3-5% of the overall running time, and a preliminary SIMD implementation shows that a 64^3 simulation runs at 130 FPS, making the computational cost of the module totally negligible.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]:

1. Introduction

Fluid simulation has found applications in visual effects in part because working with large amounts of real fluid can often be expensive and dangerous. In the case of a boiling fluid, these prohibitive factors are especially true. However, the physical process of boiling still creates intricate, visually desirable phenomena, so recent efforts in visual simulation have attempted to capture this phenomenon computationally.

Recent work [LSSF06, MUM*06, ZYP06, KLL*07] proposes extensions to the particle level set (PLS) method that are geared towards capturing this phenomena. We instead propose a modular approach, where the features specific to boiling are handled in a separate simulation, and then loosely coupled to an existing particle level set solver. This approach has several advantages, including simple implementation and high efficiency. In our experience, the module adds between than 3-5% to the overall running time. The algorithm is also highly SIMD-amenable, allowing the boiling computation to be farmed out to a GPU or Playstation 3 Cell processor. Since this computation would take place in parallel, the computational cost of our module would become totally negligible. In the case of a fully on-GPU level set fluid solver [CTL07], adding boiling also becomes straightforward. A working C++ implementation is available in the supplemental materials.

The modular design decouples bubble dynamics from water surface dynamics from both a visual and computational perspective. This makes it possible to tune the 'look' of the bubbles and the surface turbulence separately, and also allows the two features to be run at *different resolutions*, with

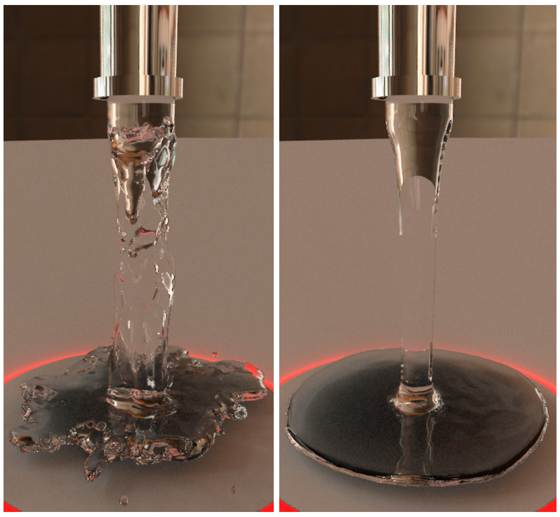
(a) With boiling module (b) Without boiling module

Figure 1: *Boiling water poured on a heating element. With the boiling module, the water jumps and pops as it hits the hot plate. A bubble automatically forms near the faucet, breaking up the (in this case) undesirable smoothness of the flow. The boiling consumed less than 5% of the running time on a 64^3 grid.*

decoupled timesteps depending on which the user deems to be the more visually relevant to their application. The module is also fluid solver agnostic, so while we use particle level set here, it could in principle be added to a Lattice Boltzmann Method [TKPR06] or Smoothed Particle Hydrodynamics [MSKG05] solver as well.

We develop a model that extends the Yanagita model of boiling [Yan92]. The Yanagita model captures the major phenomenological features of boiling using a surprisingly small number of terms. However, it notably lacks interaction with a water free boundary, and has no treatment of surface tension. The lack of surface tension results in the formation of visually disturbing ‘pancake’ bubbles, an artifact that must be eliminated to make the model visually viable. We propose methods for adding both of these features into the model while maintaining its overall simplicity.

2. Previous Work

Inspired by seminal works [FM96, Sta99, FF01], there has been a recent explosion in visual simulation of fluids. Many excellent works have been published, but for the sake of brevity, we will only focus on those that deal with boiling and bubble dynamics here. The interested reader is referred to the previous works sections of other recent work [KFCO06, ETK*07, LSSF06, KCC*06] for a more general overview.

Several different approaches have been applied to the problem of bubble dynamics. Roman [Rom01] stated the problem as one of a spring-mass system that handled surface tension and film elasticity. Kück et al. [KVG02] proposed a similar approach for bubble clusters, and proposed a rendering method for handling the thin film effects of conjoined bubbles. Hong and Kim [HK03] described a volume of fluid (VOF) method of handling the somewhat different case of bubbles rising in water. In a departure from the usual Eulerian methods, Müller et al. [MSKG05] proposed a particle-based method of simulating similar phenomena.

Bubbles coupled with an evolving free surface was handled by Greenwood and House [GH04] by using the particles in the particle level set method to detect when bubbles would plausibly form in the simulation. Hong and Kim [HK05] proposed a ghost fluid method [FAMO99] for handling the discontinuous pressure associated with bubble surfaces. Losasso et al. [LSSF06] extended the method to compute the ghost values on the fly for multiple fluid regions. Mihalef et al. [MUM*06] described a boiling simulation that departed from the usual particle level set in favor of a coupled level set / volume of fluid method in order to better capture surface tension effects. Zheng et al. [ZYP06] proposed a regional level set method to handle thin films, and described an unconditionally stable method of handling surface tension. Kim et al. [KLL*07] proposed a volume preserving extension.

Finally, we note that Harris et al. [HCSL02] have previously observed the SIMD-amenability of the boiling model we extend in this paper. They provide an excellent overview of the model, as well as extensive GPU implementation details.

3. The Yanagita Boiling Model

In this section, we give an overview of the Yanagita model for boiling [Yan92]. In its original form, the model is almost suitable for visual simulation, save for three significant drawbacks that we will address in the next section. The main strength of the Yanagita model is that it only needs to apply a handful of simple operators to a single heat field. No explicit tracking of pressure, phase, viscosity, or the air/water interface is performed, and yet the major visual features of boiling are captured.

We track a heat field T over a 2D regular grid. Extension to 3D is straightforward and left as an exercise to the reader. We denote the grid coordinates of T using a subscript thus: $T_{x,y}$. The current timestep t is denoted with a superscript: $T^t_{x,y}$. A single timestep of the Yanagita model involves two substeps, which we will denote $T^*_{x,y}$ and $T^{**}_{x,y}$.

First, we account for heat diffusion using the usual 5 point Laplacian:

$$T^*_{x,y} = T^t_{x,y} + \frac{\varepsilon}{4}(T^t_{x-1,y} + T^t_{x+1,y} + T^t_{x,y-1} + T^t_{x,y+1} - 4T^t_{x,y}). \qquad (1)$$

The symbol ε is a diffusion constant. Next, we account for buoyancy forces in the vertical direction, which we choose to be the y direction:

$$T^{**}_{x,y} = T^*_{x,y}(1 - \frac{\gamma}{2}(\rho(T^*_{x,y+1}) - \rho(T^*_{x,y-1})). \qquad (2)$$

The symbol σ is a buoyancy constant that defines the speed of the rising bubble, and ρ is a temperature-to-phase conversion function defined thus:

$$\rho(T) = \tanh(\alpha(T - T_c)). \qquad (3)$$

We note that Yanagita actually refers to this quantity as ‘density’ but we believe it more appropriately referred to as phase since it can take on negative values. The symbol α is a smearing constant that defines how sharply to resolve the phase, and T_c is the boiling temperature of water. Finally, we must account for the latent heat released by water as it converts to steam, and the heat absorbed by steam converting back to water. Yanagita offers a somewhat unconventional treatment of this, which is perhaps best stated in two steps. First, the equation itself:

$$\begin{aligned} T^{t+1}_{x,y} = T^{**}_{x,y} &+ \eta(\text{\# neighbors converted to steam}) \\ &- \eta(\text{\# neighbors converted to water}). \end{aligned} \qquad (4)$$

The symbol η is a latent heat constant. A cell converts from water to steam if its temperature crosses the boiling temperature T_c. A given cell $T_{x,y}$ converts to steam if the following is true: $(T^t_{x,y} < T_c \wedge T^{**}_{x,y} > T_c)$. Conversely, a cell converts from steam to water if the opposite is true: $(T^t_{x,y} > T_c \wedge T^{**}_{x,y} < T_c)$. Each grid cell evaluates these terms for all of its neighbors and substitutes the totals into Eqn. 4.

Since we assume that bubbles rise in the y direction, the bottom boundary is set to a fixed temperature T_{bottom} slightly less than T_c. The top boundary is set to a cooler T_{top}. For simplicity, the left and right boundaries are set to be periodic. Suitable settings for all the constants are provided in the supplemental materials.

4. An Extended Yanagita Model

There are several drawbacks to the Yanagita model that must be addressed. First, the discretization is not quite consistent, so the model is not convergent. Second, since there is no treatment of surface tension, the bubbles tend to be flat, and merge too aggressively to form ‘pancake’ bubbles (Figure 2). Third, the fluid is assumed to be in a box, with no treatment for an upper free boundary. We propose methods for addressing all of these drawbacks in this section.

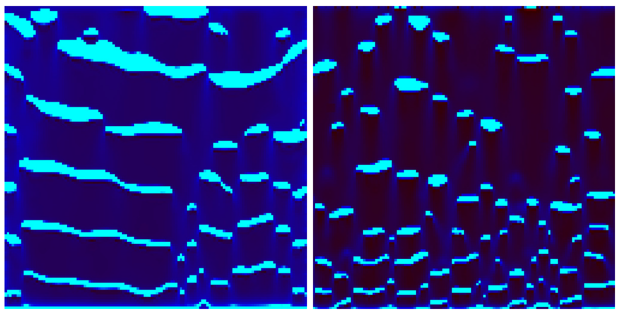
(a) Without surface tension (b) With surface tension

Figure 2: *Heat visualization of a 2D boiling simulation. Bubbles rise from the bottom towards the top.* ***Left:*** *The Yanagita model does not include surface tension, so the bubbles merge too aggressively into ‘pancakes’* ***Right:*** *Our new model adds surface tension, eliminating the pancakes.*

4.1. Convergence

In its current form, the Yanagita model is not *convergent*. Usually convergence is used to describe the rate at which successive refinements to a simulation grid will approach the true solution of the underlying PDE. Yanagita did not propose an underlying PDE that corresponded to his discretization, so we adopt a looser meaning of convergence here. The Yanagita model is not convergent in the sense that doubling the resolution of the simulation does not produce a more refined version of the same simulation.

This is relevant to graphics, since users expect increased resolution to translate to refined results, not entirely different results. In order to see why the scheme is not convergent, we first posit that the model is a discretization of the following PDE:

$$\frac{\partial T}{\partial t} = \varepsilon \nabla^2 T - \gamma T \frac{\partial \rho}{\partial y} + sign\left(\frac{\partial T}{\partial t}\right) \eta \delta(T). \tag{5}$$

First, we observe that standard graphics techniques [Sta99] can be applied to stabilize the first two terms of the right hand side. Backward Euler can be applied to the diffusion term, and $\sigma T \frac{\partial \rho}{\partial y}$ is an advection term to which semi-Lagrangian techniques can be applied. Second, we observe that Yanagita's Forward Euler discretizations for both of these terms are clearly convergent, so the problem must lie in the final term.

In the last term, $\delta(T)$ is a delta function centered at T_c. This models latent heat by injecting η heat into the system when $T = T_c$. The *sign* function flips the sign of η depending on whether the latent heat of vaporization or fusion applies. While Eqn. 4 reflects the intuition behind this term, the discretization is not consistent.

In Eqn. 4, the latent heat is released directly to the cell neighbors, while from a physical standpoint, heat would be released by the current cell, and then distributed to the neighbors by diffusion. Eqn. 4 ruins the convergence of the overall scheme because in addition to modeling latent heat, it applies an infinitely fast diffusion process to the cell neighbors. A consistent discretization would instead be:

$$\begin{aligned} T^{t+1}_{x,y} = T^{**}_{x,y} &+ \eta(T^t_{x,y} < T_c \wedge T^{**}_{x,y} > T_c) \\ &- \eta(T^t_{x,y} > T_c \wedge T^{**}_{x,y} < T_c). \end{aligned} \tag{6}$$

Using this discretization, higher resolution grids display the expected behavior of producing refined versions of coarser simulations. The model is now convergent.

4.2. Surface Tension

4.2.1. Formulation

The bubbles in the Yanagita model tend to form ‘pancakes’, because the model does not include any treatment of surface tension. In a level set simulation, the usual method of addressing surface tension is to add forces of the form $F_{st} = -\sigma\kappa(\phi)\delta(\phi)\vec{n}$ to the velocity field of a Navier-Stokes simulation, where σ is a surface tension constant, $\kappa(\phi)$ is the local curvature of the signed distance field ϕ, $\delta(\phi)$ is a delta function that is non-zero at the interface, and $\vec{n}$ is the interface normal.

However, we do not explicitly construct a velocity field in our simulation. While such a field could easily be constructed by caching the $\gamma T \frac{\partial \rho}{\partial t}$ term, we instead propose an approach that preserves the simplicity of the Yanagita model.

We posit that by introducing an extra curvature-dependant term into Eqn. 5,

$$\frac{\partial T}{\partial t} = \varepsilon \nabla^2 T - \gamma T \frac{\partial \rho}{\partial y} + sign\left(\frac{\partial T}{\partial t}\right) \eta \delta(T) - \sigma\kappa(\rho)\delta(\rho), \tag{7}$$

we can model surface tension. The term is almost identical to the usual surface tension force, but the $\vec{n}$ term has been dropped.

Intuitively, this is equivalent to F_{st}. The physical reasoning behind surface tension is that the liquid interfaces prefer planar configurations ($\kappa = 0$). Bumps and divots in the interface, respectively the $\kappa > 0$ and $\kappa < 0$ cases, are smoothed away because they are deviations from an ideal plane. The usual F_{st} term uses the $\vec{n}$ vector to induce velocities that smooth the deviates away. Our alternate term instead smoothes these deviations away directly.

4.2.2. Computation

This formulation of surface tension introduces the need to compute the curvature of the phase field, $\kappa(\rho)$. In the level set method, curvature is usually derived directly from the signed distance field: $\kappa = \nabla \cdot \frac{\nabla\phi}{|\nabla\phi|}$. We could in principle take a similar approach, but constructing a signed distance field would again be at odds with the simplicity of our overall approach.

Instead, we use a normal differencing method [IJ85, HJ87]. Instead of computing stencils over a signed distance field, we explicitly compute surface normals and then take their differences. For each grid point p, we first compute its phase $\rho(T_p)$. For each neighbor n, we also compute its phase $\rho(T_n)$. If there is a sign flip between these phases, p is on the interface. We then compute the normals at T_p and T_n, $\vec{n}_p$ and $\vec{n}_n$. The curvature $\kappa_{p,n}$ is then computed thus:

$$\begin{aligned} \text{if}(|p-n| \le |(\vec{n}_p + p) - (\vec{n}_p + n)|) \quad & \kappa_{p,n} = \frac{|\vec{n}_p - \vec{n}_n|}{|p-n|} \\ \text{else,} \quad & \kappa_{p,n} = -\frac{|\vec{n}_p - \vec{n}_n|}{|p-n|}. \end{aligned}$$

The conditional detects if the two normals are pointing towards or away from each other, and assigns the sign accordingly. If we compute $\kappa_{p,n}$ over all the neighbors and take the average, then we have effectively computed the mean curvature at p. The final result is then multiplied by σ and added to Eqn. 6 to complete the integration.

4.3. Coupling to Particle Level Set

4.3.1. Surface Dynamics and Advection

The final component to our boiling simulator is a method of coupling the results of the boiling simulation to an existing particle level set (PLS) solver. While we could attempt a complex coupling, we have found a simple approach to be effective. For any cell that is on a bubble interface, and is less than 5 cells away from the level set interface, inject a force of unit magnitude in the direction of the bubble normal. If the existing PLS simulator already has a public accessor function to its signed distance field, and provides a force application function, no modifications to the simulator are necessary. In order to ensure that bubbles are eventually absorbed into the air when they reach the interface, we set $T_{x,y} = T_{top}$ in all cells where the signed distance field is greater than some small value.

This coupling qualitatively captures two visual features. Physically, when a rising bubble approaches the water/air interface, it deforms the surface outwards. When the bubble pops, it creates a void that the water rushes in to fill, creating a splash. In our simulation, a bubble initially approaches the level set interface and pushes it outwards, as expected. When the bubble passes through the interface, the bottom hemisphere injects forces that push downwards, creating a void and splash.

We would also like the bubble to move along with the flow, so we perform advection on the heat field as well. If the fluid solver already has a generic advection function, this can be accomplished by simply calling the function on T. Otherwise, a semi-Lagrangian scheme for T can easily be implemented. The full equation we are now solving becomes

$$\frac{\partial T}{\partial t} = \varepsilon\nabla^2 T - \gamma T\frac{\partial\rho}{\partial y} + sign\left(\frac{\partial T}{\partial t}\right)\eta\delta(T) - \sigma\kappa(\rho)\delta(\rho) - (\mathbf{u}\cdot\nabla)T, \tag{8}$$

where $\mathbf{u}$ is the velocity field of the fluid solver.

4.4. Fluid Solver Advection

We use BFECC [KLLR05] for the advection of the fluid velocity field. In [DL03], Dupont and Liu take special care when the velocity field is not smooth, and only use BFECC for the velocity on grid points if the velocity is smooth in *all* dimensions. Their experiments were done in 2D on a regular grid, and we found that such a harsh test in 3D on the staggered grid we use was not necessary. However, when we did nothing, the surface would tear apart too easily. We found that locally turning off BFECC on a dimension by dimension basis worked well. For example, on a cell face in the x-dimension we use BFECC if

$$\begin{gathered} |u_{i+1,j,k} - 2u_{i,j,k} + u_{i-1,j,k}| \le \\ \min(|u_{i+1,j,k} - u_{i,j,k}|, |u_{i,j,k} - u_{i-1,j,k}|). \end{gathered}$$

4.5. SIMD Parallelization

One of the strengths of this modular design is that it allows the PLS and boiling modules to execute in parallel, at different resolutions, and with decoupled timesteps. The simplicity of the boiling algorithm makes it highly SIMD-izable, allowing complementary hardware such GPUs or the Synergistic Processing Elements (SPEs) of a Playstation 3 Cell processor to execute the module concurrently with the PLS simulation.

[HCSL02] already showed that the original Yanagita model can be ported efficiently to the GPU. Our modifications should be equally easy to port, since we have actually simplified equation 4, and the surface tension stage only requires a local grid neighborhood.

To demonstrate the ease of porting to other SIMD platforms, we instead implemented our extended Yanagita model on the SIMD units of the Cell processor. The serial version was run an Intel 3.4 Ghz Xeon, and ran at 5.35 frames per second on a 64^3 grid. We ran our Cell implementation on a IBM QS20 Cell blade, which has a total of 16 SPEs. We split the computational workload evenly between the available SPEs and measured the frames per second (FPS) as more SPEs were added. The scaling can be seen in Figure 3. We show the ideal scaling based on the FPS of one SPE for comparison. The scaling degrades as more SPEs are added because the memory bus becomes saturated, so the SPEs become starved for data.

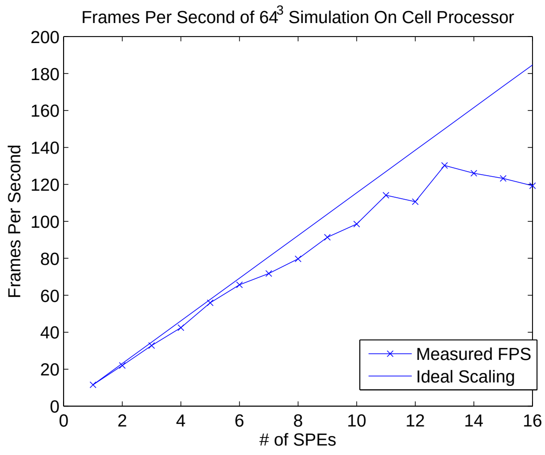

Figure 3: *Performance of a 64^3 boiling simulation on a Cell Processor. With 6 SPEs (the number in a PS3), we achieve 65.59 FPS. The performance peaks at 13 SPEs at 130.22 FPS. The scaling degrades as more SPEs are added because the memory bus becomes saturated. The CPU version runs at 5.35 FPS on a 3.4 Ghz Xeon, so the Cell offers a 24.3x speedup.*

We note that the implementation is only a prototype, and with more aggressive optimizations, even larger speedups can almost certainly be attained. Our serial implementation only consumed 3-5% of the running time to begin with, so parallel SIMD execution would make our module totally computationally negligible.

4.6. Rendering

Because of the loose coupling between the level set and boiling simulators, there is no guarantee that a bubble surface will not cause a visually disturbing interpenetration with the level set surface. However, most renderers provide a method of sidestepping this problem. The artifacts can be eliminated if a ray is not allowed to intersect a bubble surface without first passing through an odd number of level set surfaces. This information can be tracked via ray labels, a common feature of most renderers. All RenderMan compliant renderers, for example, have this feature. We used pbrt [PH04], a renderer that did not initially support ray labels, but we easily extended it to support this functionality. The light probes used in the renderings are all from Paul Debevec's High Resolution Light Probes gallery [Deb06], which he has generously made available online.

5. Results

In Figure 4 we show water boiling in a beaker, and show the viability of our coupling to PLS. The boiling module consumed less than 3% of the simulation time. In Figures 1, 5, and 6 we pour hot water onto a heating element. For comparison, we show the results of a PLS simulation without our module. As the water hits the plate, the water jumps and pops using our module, as opposed to the unconvincing smoothness of straight PLS. Additionally, a bubble forms near the faucet using our module, producing an unsteady flow that lends the results additional realism. The module consumed less than 5% of the simulation time.

Finally, to show the detail that our extended model can achieve on a relatively coarse (64^3) grid, we drop 4 hot spheres into a tank of water in Figure 7. No free surface was included in this simulation in order to draw focus to the bubbles. Note how the bubbles trail convincingly behind the spheres and continues to boil off in sheets. Without the overhead of PLS, a serial implementation runs at 2.73 frames per second; sufficient for interactive previewing of results. A full 3D SIMD implementation would almost certainly run at a real-time 30 frames per second. All the timings were performed on a 64^3 grid using a 3.4 Ghz Xeon.

6. Conclusions and Future Work

Notably, our method does not handle the case where the bubbles are advected along the free surface of the water. For this, a more sophisticated coupling to a monolithic approach [ZYP06, LSSF06, KLL*07] would be necessary. Far from the interface however, our computationally cheaper method would still be the preferred method of tracking boiling dynamics. Our module is also solver agnostic, so it should be straightforward to plug it into Lattice Boltzmann Method (LBM) or Smoothed Particle Hydrodynamics (SPH) simulations as well. However, we leave the specifics of this to future work. Finally, given the efficiency achievable with modular design for this visual feature, it begs the question of what other features can be simulated in a similar manner.

References

[CTL07] CRANE K., TARIQ S., LLAMAS I.: *GPU Gems 3*. 2007, ch. Real-time Simulation and Rendering of 3D Fluids.

[Deb06] DEBEVEC P.: *High-Resolution Light*

Probe Image Gallery. 2006. available at http://gl.ict.usc.edu/Data/HighResProbes/.

[DL03] DUPONT T. F., LIU Y.: Back and forth error compensation and correction methods for removing errors induced by uneven gradients of the level set function. *Journal of Computational Physics* (2003), 311–324.

[ETK*07] ELCOTT S., TONG Y., KANSO E., SCHRÖDER P., DESBRUN M.: Stable, circulation-preserving, simplicial fluids. *ACM Transactions on Graphics* (2007).

[FAMO99] FEDKIW R., ASLAM T., MERRIMAN B., OSHER S.: A non-oscillatory eulerian approach to interfaces in multimaterial flows (the ghost fluid method). *Journal of Computational Physics* (1999).

[FF01] FOSTER N., FEDKIW R.: Practical animation of liquids. *Proc. of SIGGRAPH* (2001), pp. 15–22.

[FM96] FOSTER N., METAXAS D.: Realistic animation of liquids. *Proceedings Graphics Interface* (1996), 204–212.

[GH04] GREENWOOD S., HOUSE D.: Better with bubbles: Enhancing the visual realism of simulated fluids. *Proc. of ACM SIGGRAPH Symposium on Computer Animation* (2004).

[HCSL02] HARRIS M., COOMBE G., SCHEUERMANN T., LASTRA A.: Physically-based visual simulation on graphics hardware. In *Proc. 2002 SIGGRAPH / Eurographics Workshop on Graphics Hardware* (2002).

[HJ87] HOFFMAN R., JAIN A.: Segmentation and classification of range images. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* (1987).

[HK03] HONG J., KIM C.: Animation of bubbles in liquid. *Proceedings of Eurographics 2003 22*, 3 (2003).

[HK05] HONG J., KIM C.: Discontinuous fluids. *Proc. of SIGGRAPH* (2005).

[IJ85] ITTNER D., JAIN A.: 3d surface discrimination from local curvature measures. *Proceedings of Computer Vision and Pattern Recognition (CVPR)* (1985).

[KCC*06] KIM J., CHA D., CHANG B., KOO B., IHM I.: Practical animation of turbulent splashing water. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2006).

[KFCO06] KLINGNER B. M., FELDMAN B. E., CHENTANEZ N., O'BRIEN J. F.: Fluid animation with dynamic meshes. In *Proceedings of ACM SIGGRAPH 2006* (Aug. 2006).

[KLL*07] KIM B., LIU Y., LLAMAS I., JIAO X., ROSSIGNAC J.: Simulation of bubbles in foam by volume control. In *Proceedings of ACM SIGGRAPH* (2007).

[KLLR05] KIM B., LIU Y., LLAMAS I., ROSSIGNAC J.: Flowfixer: Using bfecc for fluid simulation. In *Eurographics Workshop on Natural Phenomena* (2005).

[KVG02] KÜCK H., VOGELCSANG C., GREINER G.: Simulation and rendering of liquid foams. *Proceedings of Graphics Interface* (2002).

[LSSF06] LOSASSO F., SHINAR T., SELLE A., FEDKIW R.: Multiple interacting liquids. *Proc. of SIGGRAPH* (2006).

[MSKG05] MÜLLER M., SOLENTHALER B., KEISER R., GROSS M.: Particle-based fluid-fluid interaction. *Proc. of ACM SIGGRAPH Symposium on Computer Animation* (2005).

[MUM*06] MIHALEF V., UNLUSU B., METAXAS D., SUSSMAN M., HUSSAINI M.: Physics-based boiling simulation. *Proc. of ACM SIGGRAPH Symposium on Computer Animation* (2006).

[PH04] PHARR M., HUMPHREYS G.: *Physically Based Rendering: From Theory To Implementation*. Morgan Kaufmann Publishers, 2004.

[Rom01] ROMAN D.: Animation of soap bubble dynamics, cluster formation and collision. *Proceedings of Eurographics* (2001).

[Sta99] STAM J.: Stable fluids. *Proc. of SIGGRAPH* (1999), 121–128.

[TKPR06] THUEREY N., KEISER R., PAULY M., RUEDE U.: Detail-preserving fluid control. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2006).

[Yan92] YANAGITA T.: Phenomenology of boiling: A coupled map lattice model. *Chaos 2* (1992).

[ZYP06] ZHENG W., YONG J., PAUL J.: Simulation of bubbles. *Proc. of ACM SIGGRAPH Symposium on Computer Animation* (2006).

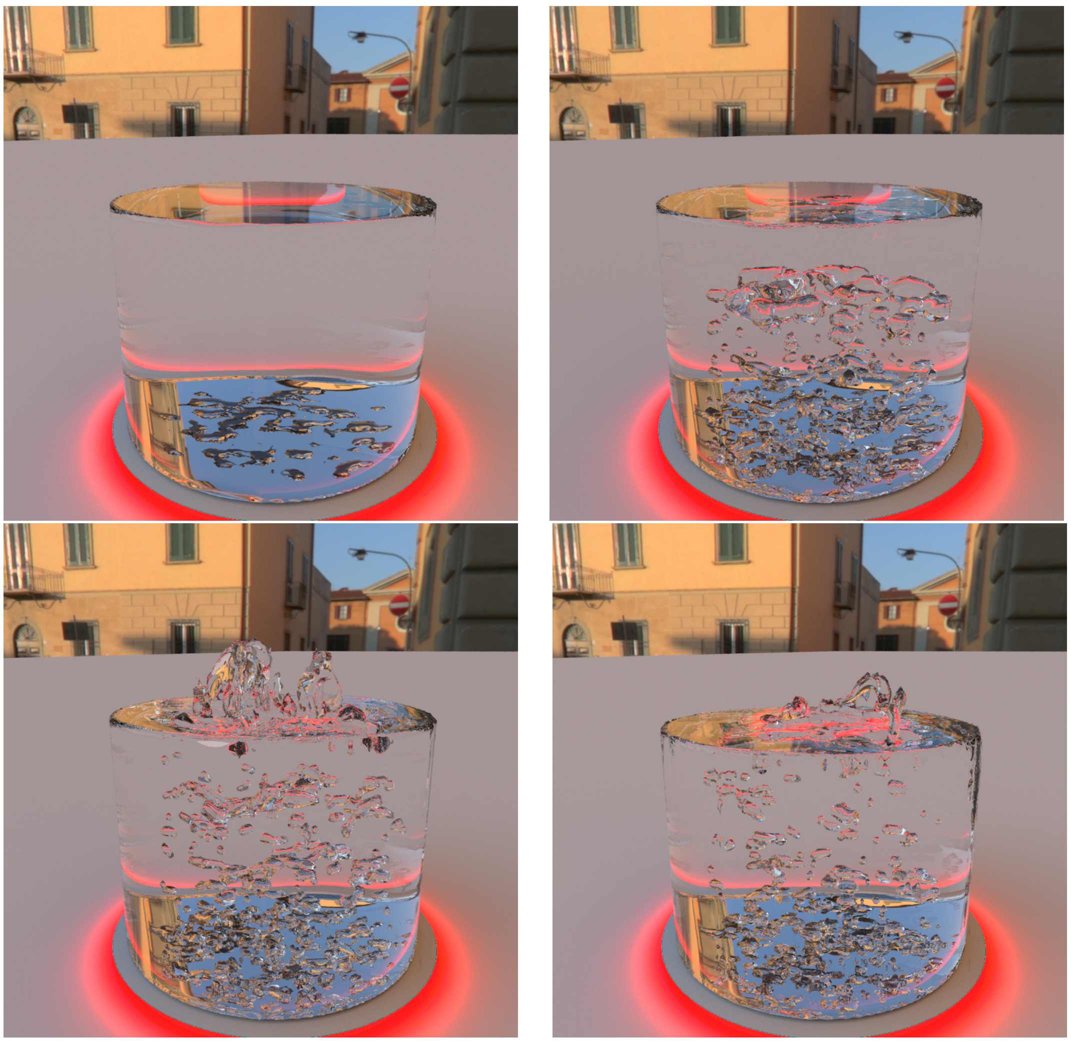
Figure 4: *Water boiling in a beaker. The splashing shows the viability of our coupling to particle level set.*

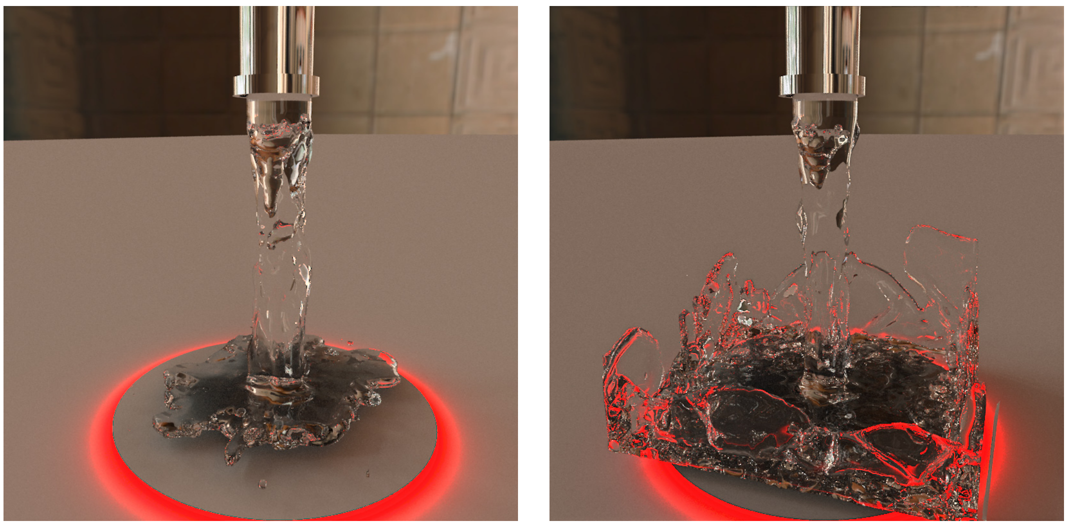
Figure 5: *Boiling water is poured on a hot plate. The water pops and jumps as it initially hits. (continues on next page)*

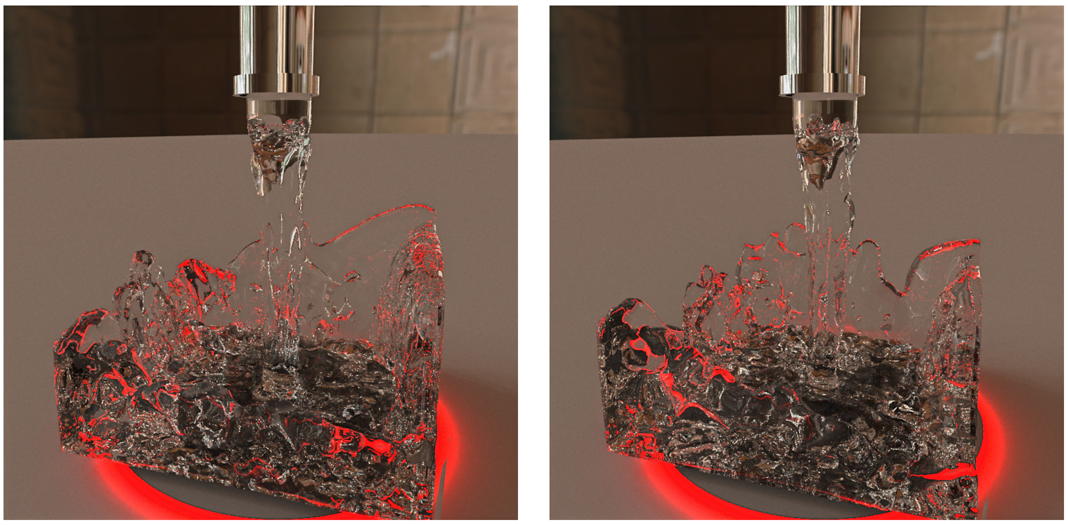

Figure 6: *As the box fills, the boiling breaks up the smooth water surface (see attached video)*

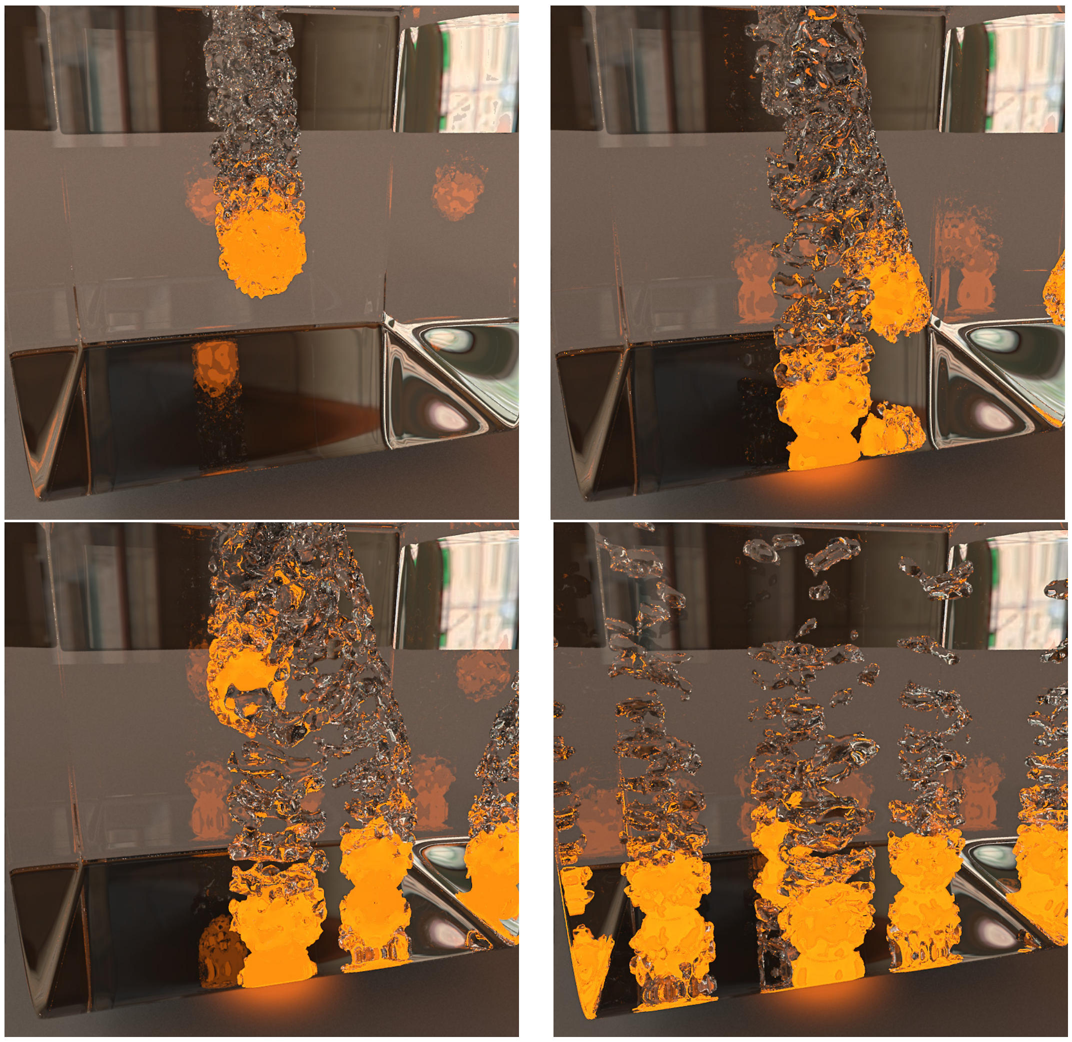

Figure 7: *Four spheres dropped into water. This* 64^3 *simulation runs at 5.35 FPS (CPU) and 130 FPS (Cell).*