Extracting and Parametrizing Temporally Coherent Surfaces From Particles

Chen Shen Apurva Shah
Pixar Animation Studios

From pouring sauces to sudsy sink water to violent sewer rapids, realistic animation of fluids presented interesting challenges in *Ratatouille*. The various fluid effects were simulated either using a physically-based solver or directly with generic particle systems. Although the simulated particles move as a whole like a fluid, the number of particles was too small to give the appearance of a continuous surface if rendered directly. To address this, we developed a technique to efficiently extract temporally and spatially coherent surfaces from particles with parametrization that allows textural details to be later added in rendering.

Figure 1: Fluids in *Ratatouille*.©Disney / Pixar.All rights reserved.

1 Extracting Implicit Surfaces

We use an implicit approach ([Zhu and Bridson 2005] and [Shen et al. 2004]) to extract fluid surfaces from particles. We build implicit functions that define signed distance fields to the fluid surfaces. For an evaluation point, our technique queries a set of particles in a certain support region then creates a reference particle with a weighted average position and radius. The signed distance from the evaluation point to the nearest point on the reference particle is assigned as its implicit function value. Using a hierarchical data structure, for example a *k*-d tree, significantly improves the spacial query and provides an efficient evaluation. This implicit function is then processed by a contouring procedure, such as marching cubes, that extracts the zero iso contour into a triangle mesh.

2 Improving Temporal Coherence

When we applied this technique to each frame of a particle simulation, we could extract a sequence of surfaces for a fluid animation. However, these surfaces were not temporally continuous. To address this deficiency, we build temporally continuous signed distance functions by blending adjacent frames. A naive way to blend signed distance functions between frames causes a severe volume loss since adjacent signed distance functions don't match in space. We used a velocity field to track the correspondences in the previous and/or subsequent frames. The velocity field can be reconstructed by weighted averaging velocity information from input particles in a certain support region. For a given evaluation point, the signed distance function values in the previous, current and/or subsequent frames corresponding to this position are retrieved and averaged.

3 Parametrizing and Tagging Properties

In order to create believable surface details, the fluid meshes needed parametrization that would allow properties like texture coordinates

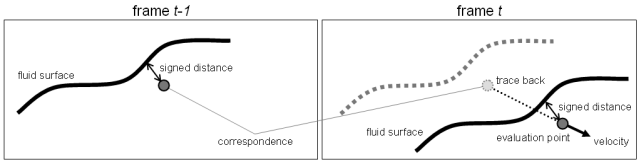

Figure 2: Improving temporal coherence

and shader program parameters to be tagged on to mesh vertices. To achieve this, we use either the simulated particles or extra helper particles, to carry the parametrization or other properties from the animation. The mesher then builds continuous functions for basic texture spaces or other tagged properties by weighted averaging of the tagged information in a certain support region. Then in the rendering step, texture coordinates or other vertex varying properties can be easily incorporated into the shader.

Since the tagging operation is handled by the mesher itself, either during the main surface generation using the same set of fluid particles or later with independent helper particles, the kind of properties that can be tagged is only limited by the imagination of the effects artists. For example, in a sauce pouring shot, we let the particles carry the local coordinates of their rest positions as texture *uvw*s [Goktekin et al.]. In another sequence, a character falls into a sink of sudsy water and generates splashes. The suds texture needed to move with the flow. Effects animators were able to use this technique to pick key frames in the simulation to project a 2D texture onto the top layer particles in the sink. In the sewer rapids shots, foam particles were simulated over the fluid flow. In a pre-rendering pass the vertices of the fluid mesh were tagged based on the local density of the foam particles. The water shader then used the foam density to create a layer of white-water on top of the rapids flow. [Froemling et al.]).

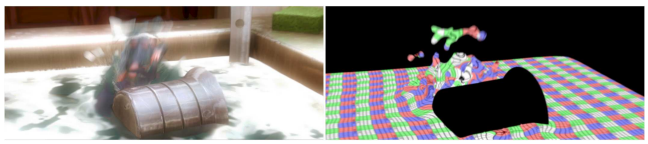

Figure 3: Tagging Properties.©Disney / Pixar. All rights reserved.

References

FROEMLING, E., GOKTEKIN, T., AND PEACHEY, D. Simulating whitewater rapids in *ratatouille*. *Submitted to Siggraph 2007 Technical Sketches*.

GOKTEKIN, T., REISCH, J., PEACHEY, D., AND SHAH, A. Rolling the dough, cracking the egg and pouring the sauce. *Submitted to Siggraph 2007 Technical Sketches*.

SHEN, C., O'BRIEN, J. F., AND SHEWCHUK, J. R. 2004. Interpolating and approximating implicit surfaces from polygon soup. *ACM Transactions on Graphics 23*, 3 (Aug.), 896–904.

ZHU, Y., AND BRIDSON, R. 2005. Animating sand as a fluid. *ACM Transactions on Graphics 24*, 3 (Aug.), 965–972.