Waterbending: Water Effects on *The Last Airbender*

Ian Sachs   Christopher D. Twigg   Lee Uren   Dan Pearson   Nick Rasmussen
Industrial Light & Magic*

In *The Last Airbender*, citizens of the Water Nation are bestowed with the ability to control water. Under a waterbender's control, water is able to take on a huge variety of shapes both furious and serene. Turning these gifts into a visually compelling reality was one of the main technical challenges faced in production of *The Last Airbender*.

1 Waterbending Pipeline

The shot process began with keyframed animations of the shapes that the water is supposed to assume. These highly choreographed animations serve as the guides for the subsequent fluid simulations. Depending on the nature of the shot, we either ran a modified FLIP/PIC simulation based on [Zhu and Bridson 2005] and [Losasso et al. 2008], or used the particle level set method (PLS) of [Enright et al. 2002] with removed particles providing bubble and splash effects. For targeting the keyframed animations, FLIP/PIC simulations could leverage any particle controls available, while PLS simulations were steered by control forces designed to push the fluid simulation toward the target shape. Commonly, removed water particles were fed into a secondary FLIP/PIC simulation for incompressible and surface tension effects. Both the PLS and FLIP/PIC simulations were run in parallel on ILM's renderfarm.

2 Fluid Simulation Enhancements

To generate control forces, we use the algorithm described in [Shi and Yu 2005]. Unfortunately, these controls became overly stiff at the magnitudes required for fast moving target objects. To assist with these fast-moving fluids, we augmented the targeting framework with "shape constraints". These additional control elements are computed in a manner similar to the control forces above, but are then explicitly blended into the velocity field rather than applied as forces. In areas where targeting needed to be exact, the fluid constraint is used as a boundary condition in the pressure projection step of the fluid solve, thereby exactly matching the control velocities in those areas. In practice, we used a combination of techniques, with shape constraints handling the "core" of the simulation, and control forces acting at the water surface.

Additionally, in the case of fast-moving fluids, it is helpful to extrapolate control velocities over the entire fluid domain. This necessitates computing signed distance everywhere as well. For this, we implemented a parallel fast sweeping method which provided a significant speedup over fast marching.

More violent effects involving a lot of splashing required the simulation of surface tension effects to capture the proper coalescing of water flying through the air. In many cases, simple inter-particle attraction forces sufficed, but for some of the smaller scale effects, more structured results were desired. These were achieved by solving for the pressure jump at the interface during the incompressibility-enforcement step in our FLIP/PIC solver.

3 Rendering

Finally, to render the resulting level set and particle data, we developed a new procedural geometry system for generating surfaces from level sets and particles, targeting both Mental Ray and RenderMan. Our system uses a node graph interface, allowing for maximum user control of the surface creation process. Graphs can consist of fully implicit volume data, sampled grid data, and explicit geometry representations such as particles and meshes. Grids are typically generated by sampling the levelset and velocity data from the PLS results, or are computed from particles using a RenderMan style blobby function. Grids can then be combined using constructive solid geometry (CSG) operations. Smoothing operators can be applied to both grid data or mesh data; grid smoothing uses simple Gaussian smoothing while mesh smoothing uses Laplacian smoothing with a volume-preserving correction term similar to [Eckstein et al. 2007]. Arbitrary user-defined attributes can be carried from the original simulation data through to final rendered mesh, and used to affect operations internal to the node graph like smoothing. As with the simulations, we were able to save on time and memory by subdividing the source grid into a number of smaller subgrids which could be rendered independently.

4 Discussion

Posed with the problem of making controllable, highly dynamic, and physically plausible water simulations, we present a simulation pipeline that provides for rapid prototyping by leveraging standard keyframe animation techniques. We propose a new method for targeting fast moving control objects, and describe the benefits of fast sweeping within the context of parallel water simulation control. We also suggest a grid-based surface tension calculation designed specifically for use with particle fluid simulation. Finally, we present a node-based mesh generation framework which addresses the issue of taking many disparate simulation sources and combining them into a renderable surface.

References

Eckstein, I., Tong, Y., Kuo, C.-C. J., and Desbrun, M. 2007. Volume-controlled surface fairing. In *ACM SIGGRAPH 2007 Sketches*.

Enright, D., Marschner, S., and Fedkiw, R. 2002. Animation and rendering of complex water surfaces. *ACM Trans. Graph. 21*, 3, 736–744.

Losasso, F., Talton, J., Kwatra, N., and Fedkiw, R. 2008. Two-way coupled SPH and particle level set fluid simulation. *IEEE Transactions on Visualization and Computer Graphics 14*, 4, 797–804.

Shi, L., and Yu, Y. 2005. Taming liquids for rapidly changing targets. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 229–236.

Zhu, Y., and Bridson, R. 2005. Animating sand as a fluid. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*.

*email:{isachs, ctwigg, luren, dpearson, nick}@ilm.com

Copyright is held by the author / owner(s).
SIGGRAPH 2010, Los Angeles, California, July 25 – 29, 2010.
ISBN 978-1-4503-0210-4/10/0007