Liquids in *The Croods*

Jeff Budsberg Michael Losure Ken Museth Matt Baer
DreamWorks Animation *

Figure 1. *Collection of different liquid scenarios from the animated feature: close character interaction in clear visibility, complex underwater phenomena, and a large-scale whitewater splash.*

Abstract

The Croods' world is under duress and presents the characters with a plethora of grandiose obstacles at every turn. After a torrential downpour, the Croods family is left stranded in the middle of an expansive ocean, forced to swim out of trouble. The art direction proved difficult, due to the clear tropical water and characters in fur-covered outfits. In this talk, we cover the technology and techniques for executing a challenging liquid sequence.

Keywords: animation, liquids, production, simulation, vdb, water.

1. Primary simulation

Overall, our approach was to design a main liquid simulation near the characters, embed it in the middle of a procedural ocean, and layer a succession of secondary elements (splashes, aeration, ripples, etc). We made a concerted effort to break problems down into more simplified systems and tools, which empowered artists to quickly iterate on more manageable pieces of the system.

The main liquid simulation was performed in Naiad. As typical with production character meshes, our input models contained self-intersections and rendertime geometry generators (hair/fur). We converted to narrow-band VDB [Museth 2013] level sets and back to an adaptive polygonal mesh in order to remove unwanted interior geometry, retain high fidelity, and provide clean, water-tight collision meshes.

2. Rasterize particles into VDB level sets

We decided against meshing in Naiad, due to relatively slow serial execution, and instead opted for a time-independent solution with our new VDB-based Houdini toolkit. However, we could simulate far more particles in Naiad than we could bring into Houdini. As such, we derived an importance metric in order to cull unnecessary particles, but preserve the maximum resolution at the interface. In Houdini, the footprints of the liquid particles are rasterized into a high-resolution, sparse level set. We opted for spherical footprints for speed, though in practice artists preferred velocity-aligned teardrop shapes that are composed of a multitude of advected spheres with attenuating radii. This conversion process is fully multithreaded and our 8-core workstations could process on the order of a few million particles per second depending on the particle velocity and size. We also found that VDB allowed artists to generate level sets at far higher resolutions, per significantly smaller memory footprints, than existing third-party tools based on conventional dense or tiled data structures.

* e-mail: {jeff.budsberg, michael.losure,
ken.museth, matt.baer}@dreamworks.com

3. Filtering / Morphological operations

While the rasterization of particles proved to be very fast, it only serves as an initial surface that requires post-processing to obtain the desired artistic look. Visually, an artist would want to selectively accentuate peaking and sharp features, remove artifacts, fill holes, and smooth flat areas, for example.

We performed this post-processing by means of various combinations of costume filtering and morphological operations. This unique workflow allows artists to carefully art-direct the final look instead of relying on slower and more complex monolithic turn-key solutions that attempt to produce the final high-quality surface of the particles in a single step. Instead, our artists had fast turn-around with small incremental steps that could easily be modified, reverted, and precisely sculpt the desired look.

Figure 2. *Surfacing of the base simulation, using a VDB level set as an intermediate representation. This permits user-controlled morphological operations for art direction, followed by fast adaptive meshing.*

In practice, the VDB toolset is exposed as a collection of Houdini nodes (SOPs) that allows artists a significant degree of flexibility in combining the various techniques to manipulate the level set surface. This toolset for processing the level set surface can be grouped in three types of techniques. The first group of tools is

based on morphological operators like dilation, erosion, closing and opening. They effectively allow the artist to fill holes, remove small isolated particles, and sharpen, peak or blur surface details. Next, are smoothing operators that are based on higher-order differential properties of the level set. This includes mean-curvature flow and Laplacian smoothing that perform second-order smoothing operations. Finally, we use various types of kernel-based convolution filters that can smooth (or sharpen) surface details in a very computationally efficient way. Examples hereof include Gaussian, mean-value and median-value 3D filters.

4. Level set to mesh conversion

After the high-resolution and quality level set surface was produced, we applied an adaptive dual-contouring algorithm to tessellate the surface with a mesh that best captures surface details while limiting the polygon count. Since the Naiad simulation domain only covered a portion of the ocean surface, it was tricky to maintain a seamless blend across the boundary. The naïve approach is to deform the large ocean plane with a procedural ocean shader in Houdini, merge it with our liquid simulation, and bake out a heavy per- frame polygonal model. Unfortunately, this is computationally expensive, has a very large footprint on disk, and the resolution would likely never be high enough for the movie's close-up shots.

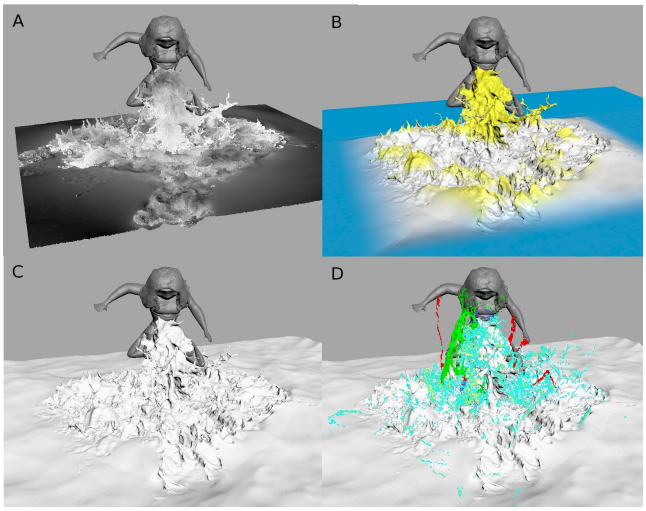

Figure 3. *Breakdown of the mesh generation. Liquid simulation as particles (only displaying particles near interface) (A); numerical analysis dictates procedural ocean displacement application, no displacement (yellow), deferred until rendertime (blue) or baked into levelset (white)(B); seamless application of ocean displacement (C); Additional geometric elements per artistic demand, driven via simulation analysis (D)*

Instead, we developed a hybrid approach to selectively apply (or defer evaluation of) the ocean displacement shader. Clearly, it is desirable to apply rendertime deformation to the ocean outside the simulation domain. However, inside the domain proves to be more complicated. Some ocean shader deformation must be applied inside the domain, otherwise it is evident that there is no correspondence to the outside motion. However, this deformation cannot be applied uniformly, as collisions would no longer be accurate and ballistics would exhibit strange mid-air behavior.

Therefore, the effect of the ocean shader was reduced or eliminated in regions we identified via a weighted metric (level set value, curvature, ballistics probability, obstacle proximity, etc). Further, it was important to bake in this deformation to the primary simulation level set, as every subsequent processing stage depended on an accurate representation of the final geometry (foam riding the surface, splash re-entry collisions, ripples, etc).

To resolve the seam, we selectively flattened the outskirts of the simulation domain to match the un-deformed ocean plane. We stored the flattening delta and importance mask as polygonal vertex attributes in order to selectively apply the procedural ocean shader at rendertime. Ultimately, we found this method very successful; we retained our hard-earned simulation detail, the simulation was successfully embedded in the procedural ocean, and we could apply rendertime deformation at virtually infinite resolution.

Additional data-driven rendertime deformations furthered the complexity. We found that no simulator could capture the regions of very high visual frequency that occur on the surface after severe disturbance; hence we amplify the turbulence artificially. We computed a temporal decay of the vorticity at the interface (calculated via VDB operators, stored as another vertex attribute) in order to drive regions of micro-detail displacement. Also, a shallow-water solver assisted in providing action to unify the simulation domain and the procedural ocean.

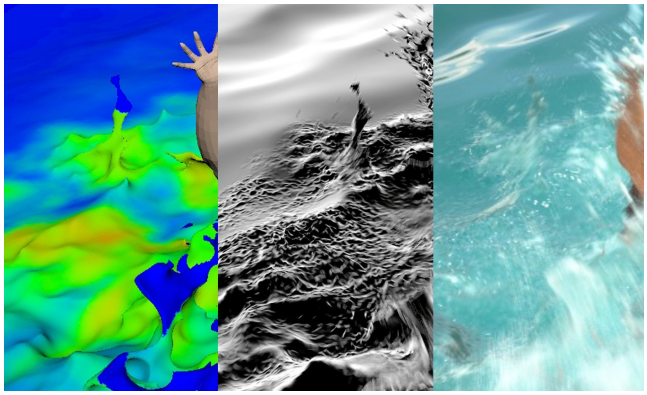

Figure 4. *Additional micro-detail displacement added at rendertime, driven by a temporal vorticity heuristic.*

5. Ballistic splashes

We did not try to capture the entire behavior in one liquid simulation, and secondary elements were completely decoupled from the main simulation for maximum artistic control. We analyzed the level set and velocity fields for important features (curvature, local velocity deviation, divergence, computed via VDB operators) in order to dictate the emission into wedged particle/SPH ballistics simulations. Artists responded well to this workflow, as they could quickly turn around high-detailed particle-tendril simulations that were catered to each shot's art direction (as evidenced by character sheeting, dripping hair, etc), and they were not bound to a restrictive fluid simulation domain.

This served us well to marry the ballistics and main simulation; as upon splash re-entry, we emitted further ballistics in the air, as well as utilized our 2D ripple solver to provide high frequency details in concert with the splash. Near the interface, we injected bubbles and particle foam using similar event-driven triggers. On the surface, we transitioned into clustering bubble dynamics, via constrained advection in VDB fields and local neighborhood surface tension.

Further, since ballistics took up far less spatial real estate than the main simulation, they could be rasterized into significantly higher-resolution VDB level sets. Morphological/filtering operations are very important with ballistics (to fill gaps/remove lumps), and we devised a successive resolution schema to maximize detail prior to extracting the adaptive mesh.

6. Volumetric elements

Highly turbulent events in the water (or air) triggered volumetric churn and aeration (or mist), which we simulated in a fast gas solver [Henderson 2012] with divergence control, explicit fluid velocity, and low dissipation. Our gas toolset allowed for precise control over every aspect of the simulation: easy flow fields, creative control over emission, and even directly via extracting/reinserting grids mid-simulation-step. For medium-scale texture, bubbles were advected through the velocity fields (often blended with other fields for control). We would often adjust bubbles post-simulation in relation to the stereo camera rig, to meet artistic goals and compose the space's 3D composition.

Many additional environmental volumetric elements were crafted in collaboration with Lighting. To capture the mysterious haze typical of underwater visibility, we created location-based volume setups to assist in art-directing light penetration, depth attenuation and hue-shifting, as well as artistic vignetting of the frame. Further, we provided a flexible shader approach to underwater god-rays, as that effect proved to be very camera specific. Of course, no water would be complete without “crap in the water”; millions of suspended particles and tiny plankton all at the whim of the ocean’s currents, generated at rendertime.

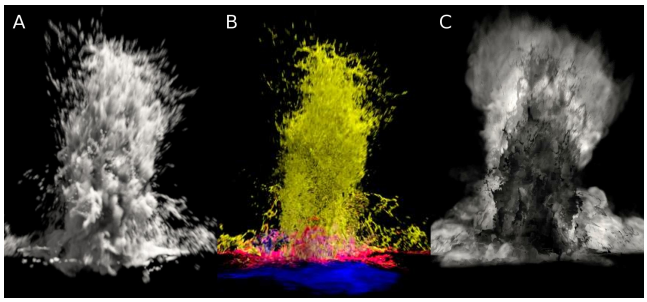

Figure 5. *Examples of volumetric elements: whitewater (A); refracted aeration (yellow), foam (red), underwater churn (blue); gaseous mist (C)*

7. Lighting and integration

All rendering was done in DreamWorks’ Reyes-based renderer via a multipass solution and ultimately composited in Nuke. Geometric elements included the main water mesh, ballistics mesh (and other secondary meshes), large underwater bubbles, crap in the water, and tiny droplets as particles. Volumetric elements included surface foam, aeration, near-interface bubbles, aerial mist, underwater haze, silt, god-rays, and whitewater (suspended inside the meshes). We made much use of simulation data in vertex attributes and extra volume fields for our shader networks per flexibility in compositing. Rendering VDB volumes proved very successful, as the hierarchical data structure is both fast and memory efficient. FX prototyped the majority of the shader networks, and Lighting/FX worked in unison for the final composite.

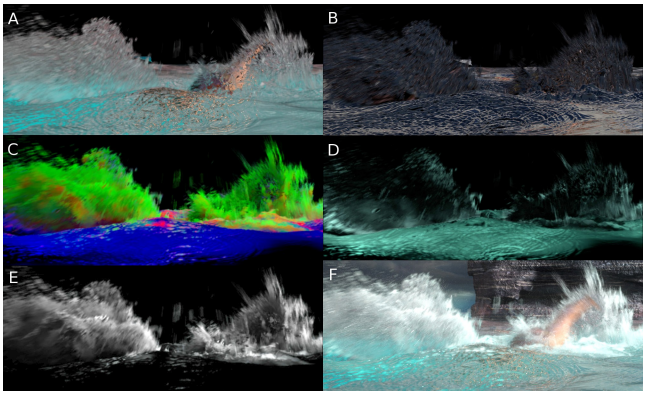

Figure 6. *A sampling of render passes that comprise the effect: graded refraction (A), graded reflection (B), refracted volumetric elements (C), graded aeration (D), graded whitewater (E), final composite (F)*

References

HENDERSON, R. Scalable fluid simulation in linear time on shared memory multiprocessors. 2012. *ACM DigiPro Symposium.*

MUSETH, K. VDB: High-resolution sparse volumes with dynamic topology. 2013. *ACM Transactions On Graphics* 32, 3.