Pahi: A Unified Water Pipeline and Toolset

Alexey Stomakhin
Wētā Digital
United States
st.alexey@gmail.com

Steve Lesser
Wētā Digital
New Zealand
slesser@wetafx.co.nz

Joel Wretborn
Wētā Digital
Sweden
joel@wbn.se

Sean Flynn
Wētā Digital
New Zealand
sflynn@wetafx.co.nz

Johnathan Nixon
Wētā FX
New Zealand
jnixon@wetafx.co.nz

Nicholas Illingworth
Wētā FX
New Zealand
nillingworth@wetafx.co.nz

Adrien Rollet
Wētā FX
New Zealand
arollet@wetafx.co.nz

Kevin Blom
Wētā FX
New Zealand
kblom@wetafx.co.nz

Douglas McHale
Wētā FX
New Zealand
dmchale@wetafx.co.nz

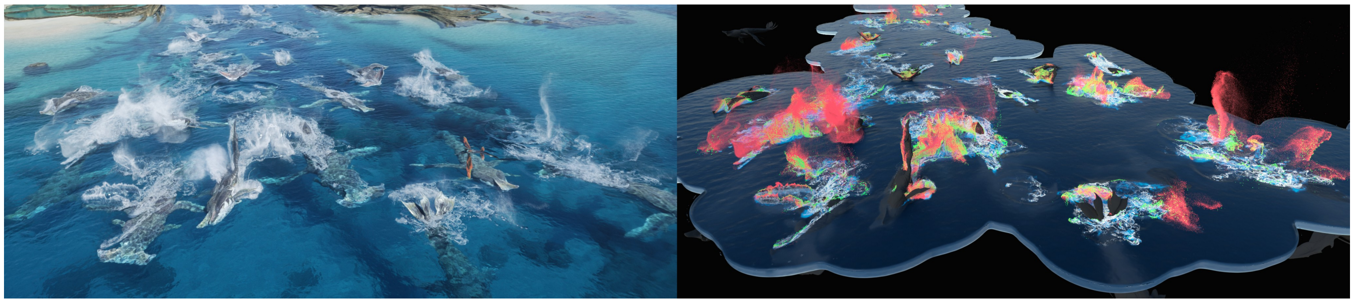

Figure 1: A shot from *Avatar: The Way of Water* (top) completed with Pahi through a distributed simulation and a breakdown of different water components (bottom): bulk fluid surface, bubbles (blue), foam (white), spray (green), and mist (red). ©Disney and Wētā FX.

ABSTRACT

We present Pahi, a unified water pipeline and toolset for visual effects production. It covers procedural blocking visualization for preproduction, simulation of various water phenomena from large-scale splashes with airborne spray and mist, underwater bubbles and foam to small-scale ripples, thin film and drips, and a compositing system to combine different elements together for rendering. Rather than prescribing a one-size-fits-all solution, Pahi encompasses a number of state-of-the-art techniques from reference engineering-grade solvers to highly art-directable tools. We do a deep dive into the technical aspects of Pahi components and their interaction, and discuss practical aspects of its use on *Avatar: The Way of Water*. We were honored to be awarded the Visual Effects Society (VES) 2023 Emerging Technology Award for this work.

CCS CONCEPTS

• **Computing methodologies → Physical simulation**; *Simulation types and techniques.*

KEYWORDS

water, simulation, pipeline

ACM Reference Format:
Alexey Stomakhin, Steve Lesser, Joel Wretborn, Sean Flynn, Johnathan Nixon, Nicholas Illingworth, Adrien Rollet, Kevin Blom, and Douglas McHale. 2023. Pahi: A Unified Water Pipeline and Toolset. In *The Digital Production Symposium (DIGIPRO '23), August 05, 2023, Los Angeles, CA, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3603521.3604291

1 INTRODUCTION

Avatar: The Way of Water required 2,225 water shots ranging from lingering close-up character interaction to frenetic open-ocean vehicle and giant creature chases, such as the one shown in Figure 1. Water development began in 2017, with the establishment of the *Water Development Project*. We treated this project as a miniature show consisting of challenging test shots, reference shoots, and dailies, all built to represent the upcoming water challenges. Rather than sprinting to deliver these shots, we used the project to evaluate and continuously refine the water pipeline, to ensure the toolset could support a crew rapidly onboarding to deliver thousands of similar shots with consistent industry-leading fidelity.

The Water Development Project found no single water solver in FX would work universally; we needed a new water toolset to give artists options from high-precision physics-based simulators to intuitive artistically driven components.

2 PREVIOUS WORK

Water is a fascinating and captivating phenomenon, which always attracts viewer attention. Various tools for creating faithful water effects have been developed and advanced by hundreds of academic and industry researchers for decades.

While there is no shortage of literature on specialized water solvers and techniques, there appear to be significantly less published materials on how to build a holistic production water pipeline. Garcia et al. [2016] provides insights into procedural ocean rigs and Palmer et al. [2017] describes render graph compositing, both used to deliver Disney's *Moana* (2016). Stomakhin and Selle [2017] shows how fluxed animated boundaries (FAB) driven by procedural Stokes/Gerstner waves can be used for open-ocean FLIP fluid simulations. SideFX Houdini and Autodesk Bifröst are current state-of-the-art frameworks encompassing large collections of water tools, which studios adapt and pipeline according to their needs. While we were able to utilize some existing techniques and technology, there was a requirement to build custom tools to ensure we had a robust, scalable, end-to-end water pipeline which we describe below. Many of the underlying motivations and reasons for such a requirement are discussed in detail in [Lesser et al. 2022; Wretborn et al. 2022].

3 ARCHITECTURE OVERVIEW

We split the major stages of water work on *Avatar: The Way of Water* into three phases: first we developed a common input format for client deliveries to the studio, second we refined it with the simulation tools, and third we rendered it into final pixels. We refer to this overall water architecture as *Pahi* [1]. Pahi focuses on providing artists with a modular toolkit for tackling their shots using a wide suite of in-house and third-party components, with near-automatic support for incorporating the results back into a format usable by the next stage. The rest of this section focuses on the high-level stages of Pahi. Section 4 provides more detail on individual components. Section 5 discusses the use of Pahi in production.

3.1 Common input

Water started with preproduction and on-set performance capture using a real-time, physically based, GPU ocean-spectra deformer (see [Horvath 2015]), built for the stage team at a client studio.

Previously, client water handover was often very approximate, using techniques such as manually deformed planes or animated cylinders as a stand-in for the desired wave heights and timings. Non-physical wave indicators limited the ability to use fluid solvers

[1]The word "pahi" is Tahitian for a large seagoing Polynesian ship with two connected hulls. Pahi's original development started with the core ideology of seamlessly connecting the two hulls of our simulation pipeline, Houdini and Loki [Lesser et al. 2022], to form one toolset for artists to use.

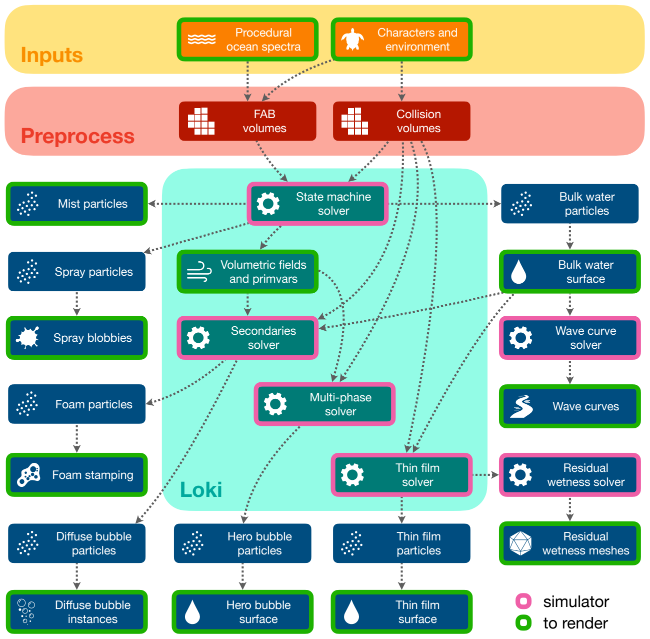

Figure 2: Overview of different Pahi components, with an example data flow between them representative of a typical open-ocean shot. ©Wētā FX.

to their full potential, as physical water would be unable to move at the given speed and height, leading to many layers of additional warping or other manipulations to match the clients' waves.

By giving the client a new procedural water tool as a physical starting point for water, we ensured that subsequent work that built on client water would be able to use state-of-the-art fluid solvers and still respected the original intent. We found that using ocean spectra provided a good balance of controllability and performance. The real-time ocean-spectra deformer gave the client a better early indication of lighting, letting them compose their shots on stage, accounting for water's interaction with light, and resulting in fewer surprises in the transition to the fully rendered delivery.

We also used the spectra to run simple buoyancy sims for characters, to give animators a physically plausible starting point.

3.2 Flexible simulation

Once the client handed over the ocean spectra to the VFX studio, the artists' tooling options rapidly expanded out to a new and improved suite of water solvers. The huge variety of water shots required a flexible approach for the FX department to turn client water into the final photoreal deliverable. Many different kinds of simulations needed to be mixed and layered using numerous different water tools, ranging from direct usage of the client wave spectra, to distributed FLIP simulations, to specialized components such as wave curves or anisotropic surfacing (see Figure 2). The simulation framework Loki [Lesser et al. 2022], with a focus on coupling, state transitions, and distributed computing was the centerpiece for most of these solvers.

Usually the outputs of Pahi would be Loki compositing graphs, which are networks of volume and particle compositing built to combine the simulation results in a spatially varying and lossless form. These Loki compositing graphs could be sampled for further simulations, or published and passed to Lighting to be rendered.

3.3 Renderables

The Manuka renderer [Fascione et al. 2018] generated the final pixels by sampling Loki compositing graphs (which we call render graphs) from Pahi. The goal of these render graphs is to seamlessly and losslessly combine the results from the wide variety of water tools the Pahi iterations used to bring the water together into a cohesive whole. This lossless combining process was particularly important since the water tools were generally run at a variety of resolutions, leading to some components such as wave curves outputting very fine detail, while other components such as bulk simulations or wave spectra providing more coarse detail or completely procedural data. Rather than discretizing everything together into a uniform format such as a VDB [Museth 2013], Loki render graphs, such as the one shown in Figure 3, combine the various components per particle or volume sample; and they can load and sample each data source at their original resolution. This avoids the need to select a single uniform voxel size, and supports the modular approach of Pahi to mix and match components at will without the need to worry about combining multiple layers, which can lead to degradation in quality.

4 PAHI COMPONENTS

4.1 Bulk water

The cornerstone of our simulation pipeline was a FLIP-based *bulk water* solver, supporting all major state-of-the-art features including variational pressure projection [Bridson 2015], variational viscosity [Batty and Bridson 2008], surface tension [Stomakhin et al. 2019], lossless transfers between Lagrangian particles / Eulerian grid [Fu et al. 2017], and narrow-band representation [Ferstl et al. 2016]. Emission, sinking, and open-boundary ocean domains were set up using FABs [Stomakhin and Selle 2017].

While surface tension, $\sigma = 0.072\,\mathrm{N/m}$, and viscosity, $\mu = 10^{-3}$ Pa·s, of water have little-to-no effect on the fluid dynamics when simulated at voxel sizes of about 1 cm or larger (which we verified experimentally) and can be disabled for efficiency, they play a prominent role at sub-centimeter scales, leading to formation of thin film sheets, tendrils, and oscillating droplets (see Figure 4).

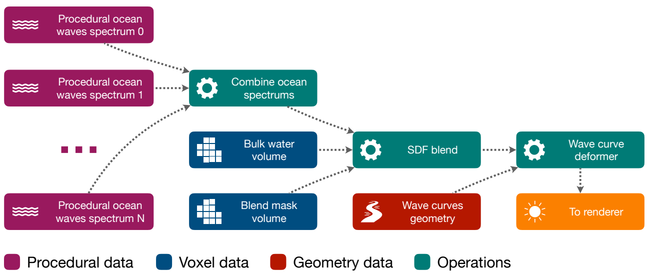

Figure 3: Example simplified render graph demonstrating creating and mixing procedural open-ocean waves, loading a surfaced bulk fluid cache from disk, blending the two, and deforming by wave curves. Each operation in the render graph is performed per sample requested by the renderer to ensure no intermediate discretizations are losing detail throughout the process. ©Wētā FX.

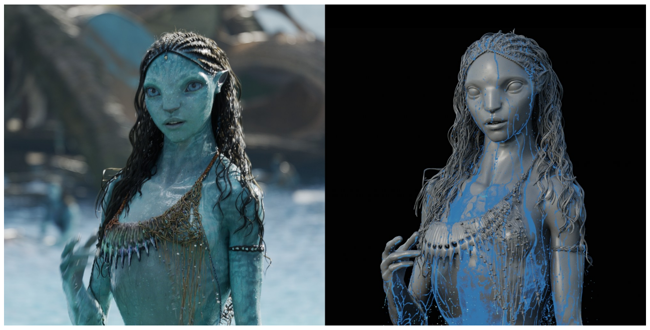

Figure 4: Thin film and drips on a character walking out of the water, simulated using our bulk water solver at voxel size 0.6 mm with surface tension and viscosity enabled. ©Disney and Wētā FX.

4.2 Loki state machine

A basic FLIP solver models water dynamics in a vacuum, which is typically sufficient to achieve realistic looks at smaller scales ($\lesssim$1 m splashes) and moderate velocities ($\lesssim$1 m/s). However, at larger scales and higher velocities, air dynamics start to have a prominent effect on water behavior and need to be taken into consideration. To efficiently capture these large-scale scenarios while still providing the essential small-scale details such as droplet interactions driven by surface tension, we adopted a multi-scale *state machine* approach, as briefly outlined in the Applications section of [Lesser et al. 2022]. In what follows we describe how we expanded upon this approach along with practical considerations for delivering a large VFX production using this technique.

A typical large-scale shot, such as the one shown in Figure 5, would be run with a voxel size of 5-20 cm. To efficiently capture the essential small- and large-scale water features simultaneously, we used multiple solvers that excel at distinct scales, run in tandem, with water automatically transitioning between the solvers while conserving mass and momentum, all coupled with air, and completed in a single pass.

Our state machine approach simulates the dynamics of water in five different states. We will focus on the airborne states in this section and the bubbles and foam states in Section 4.6.

- **Bulk water** represents the coarse volumetric motion of water as it interacts with (typically) large-scale fast-moving colliders. We simulated bulk water as per Section 4.1.
- **Spray** consists of sub-voxel droplets, represented as particles, that split off from the bulk water. Spray droplets are still large enough to interact with each other to form irregular shapes and tendrils. To capture this interaction we utilized smoothed particle hydrodynamics (SPH).
- **Mist** is composed of even smaller droplets that form from spray as it breaks up and splits. Due to their size they are incapable of forming connected structures, and we simulated mist as ballistic particles only interacting via binary droplet collisions (BDC) [Jones and Southern 2017].
- **Bubbles and foam** represent air pockets submerged in water. We simulated them using the method described in [Wretborn et al. 2022], either together with the other water states or in a separate pass for additional control.

We provide a schematic breakdown of our water state machine in Figure 6. We will reference the circled numbers that label the transitions in this figure as we describe the flow of data below.

4.2.1 Transition splitting. As liquid transitions from bulk water to the spray ① and mist ②③ representations, we increase the effective resolution by splitting the particles into smaller ones. To split a particle, referred to as a "parent", we delete it and then emit new particles repeatedly until the parent's volume is accounted for. To compensate for any small differences between the total volume of the split particles and parent volume, related to random sampling, we inflate or deflate the split particles uniformly such that the volume is conserved exactly. Transition splitting is important because it allows the sub-grid liquid dynamics to be captured with finer details. Splitting particles was an intuitive choice because it is an approximation of the real-world phenomena of airborne water breaking up into droplets and thin features, as external forces such as wind act upon it.

The radii of the new particles are determined by sampling a power-law probability density function

$$\Pi(r) = \begin{cases} \dfrac{A}{r^{\eta}}, & \text{for } r \in [r_{\min}, r_{\max}], \\ 0, & \text{otherwise,} \end{cases} \tag{1}$$

where $r_{\min}$ and $r_{\max}$ are the smallest and the largest split particle sizes allowed, respectively, η is a parameter that allows the user to control the balance of small and large droplet splits, and A is the normalization constant determined from $r_{\min}$, $r_{\max}$, and η. We found $\eta = 2$ to work well in practice. Initializing spray and mist particles in this distribution ensured that we avoided the unnatural uniformly sized ballistic particles that result from a standard FLIP simulation. We chose to use this particular probability density function, originally formulated to capture bubble distributions in breaking waves [Deike et al. 2016], because we were already using it for bubble emission; and we found that, with a smaller value of η (we typically used 2.7 for bubbles), it provided the spray and mist distribution a good starting point which was then enhanced during simulation with SPH and BDC, designed specifically for water droplet interactions.

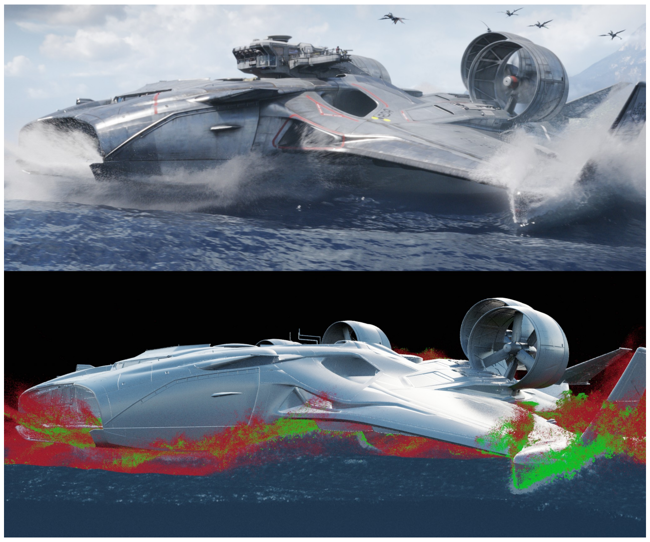

Figure 5: Frame of a shot (top) demonstrating interaction of water with a 116 m long vehicle simulated using our state machine approach. The different states are color coded (bottom): bulk (blue), spray (green), and mist (red). ©Disney and Wētā FX.

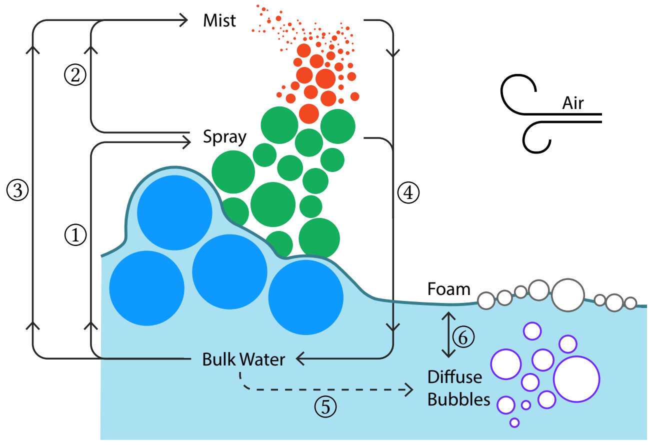

Figure 6: Schematic representation of the flow of data in our water state machine. Solid lines indicate transitions between states. As heuristic thresholds (Section 4.2.2) are met, bulk water (blue) splits into spray (green), which splits further into mist (red). Diffuse bubbles (outlined in purple) are emitted (dashed line) based on aeration (Section 4.6), and become foam particles (outlined in gray) as they reach the bulk water surface (teal). Each state is coupled with the fluid that surrounds it (air or water). ©Wētā FX.

Our users have found tuning $r_{\min}$ and $r_{\max}$ difficult, since depending on how those related to the radius of the parent particle r_{parent}, there would often be either too many splits, leading to memory overflows, or no splits at all. To provide a more straightforward way to control the radius distribution, we instead allowed users to directly specify the target characteristic number of splits k, as well as the split ratio $\xi = r_{\max}/r_{\min}$. We would then determine $r_{\min}$ from

$$r_{\min}^3 = \frac{r_{\text{parent}}^3}{k} \frac{4-\eta}{1-\eta} \frac{\xi^{1-\eta} - 1}{\xi^{4-\eta} - 1}, \tag{2}$$

and $r_{\max} = \xi r_{\min}$ per parent particle, and proceed with splitting as previously described. For the derivation details of equation (2) we refer the readers to Appendix A.

We allowed the target number of splits k to be set per particle based on the particle's transition weight $w \in [0, 1]$, which we originally envisioned to be determined from local dynamic properties of the fluid, which we would linearly remap to a global user-defined range $[k_{\min}, k_{\max}]$. By default we used $k_{\min} = 2$ in all scenarios; $k_{\max} = 32$ for transitions from bulk water to spray ① and mist ③, and $k_{\max} = 12$ for transitions from spray to mist ②. At times we reduced $k_{\max}$ from these defaults to decrease the memory footprint of very large simulations. Our users have experimented with multiple different heuristics for computing transition weights w, and eventually settled on a simple clamped linear remapping of parent particle velocity magnitude from a user-defined range $[v_{\min}, v_{\max}]$ to $[0, 1]$. We used a default velocity range of $[0, 20]$ m/s.

We determined the position of each split particle by uniform random sampling of a point in a sphere centered about the parent particle with a radius equal to $r_{\text{parent}}\beta$, where β is a user-defined multiplier used to control how spread out splits are as they are emitted. We found the value $\beta = 2$ to work well across the majority of the shots. For split particles emitted into the spray state ①, we encountered issues when particles would emit in overlapping configurations. SPH forces would push particles apart upon emission, which would break up the important spray droplet and tendril structures, and, without enough substeps, cause instabilities. To avoid this we would iteratively push particles apart from one another until there were either no overlaps, or to a maximum number of iterations $i_{\max}$. We used $i_{\max} = 10$, but found that generally it only took 1-3 iterations to remove all overlaps.

The velocities of split particles were determined differently depending on the source state of the transition. For particles transitioning from bulk water ①③, the split particle velocity was tri-linearly sampled from the bulk water velocity field. For particles transitioning from spray ②, the velocity was determined by computing an inverse distance weighted average of the nearby particle velocities. Users could also optionally add jitter to the velocities, but this was rarely needed in practice.

4.2.2 *Transition heuristics.*

We computed heuristics on the water at each state to determine when the data would transition from one state to the next, based on user-defined thresholds. We again refer to the circled numbers in Figure 6 as we describe the heuristics. The summary of all transition thresholds with recommended default values is presented in Table 1.

Bulk water particles transition to the spray particles ① when all of the following conditions are met

$$\alpha > T_s^\alpha, \tag{3}$$

$$\phi < T_s^\phi, \tag{4}$$

$$\psi < T_s^\psi, \tag{5}$$

$$v > T_s^v, \tag{6}$$

$$n < T_s^n \vee \|\nabla p\| < T_s^p, \tag{7}$$

where α is the particle's age in seconds from when it was emitted into its current state, ϕ is the distance to the bulk water surface, ψ is the distance to the nearest collider surface, v is the velocity magnitude of the particle, n is the number of particles in the containing voxel divided by the average particle-per-voxel density of the bulk water on emission (which is 8 for a typical FLIP implementation), and p is the pressure of the bulk water at the current particle position. When the pressure gradient magnitude at a particle position is low, it indicates a region of the bulk water that is nearly in free fall. We found this heuristic to be the most indicative of regions that should transition to spray. The other heuristics functioned mostly to prevent problematic transitions, like bulk water becoming spray deep below the surface, and also to provide users more control over transitions. $T_s^\cdot$ represent user-defined thresholds for each of the quantities.

Table 1: Summary of our state machine transition heuristic thresholds. The defaults were determined by our users after testing on a variety of water scenarios. We found that they generalized well across many different scales and resolutions.

Symbol	Default	Description
T_s^α	.05 s	Spray age threshold
T_s^ϕ	-2 cm	Spray to bulk-distance threshold
T_s^ψ	0 cm	Spray to collision-distance threshold
T_s^v	2.5 m/s	Spray speed threshold
T_s^n	.35	Spray density threshold
T_s^p	.5 kg/(cm·s)^2	Spray pressure gradient magnitude threshold
T_m^α	.1 s	Mist age threshold
$T_m^{\phi^c}$	0 cm	Mist to collision-distance threshold
T_N	2	Mist neighbor count
T_m^v	5 m/s	Mist speed threshold
T_b^α	.1 s	Back to bulk age threshold
T_b^ϕ	0 cm	Back to bulk distance threshold

Spray particles transition to mist particles ② when

$$\alpha > T_m^\alpha, \tag{8}$$

$$\psi < T_m^{\phi^c}, \tag{9}$$

$$N < T_N, \tag{10}$$

$$v < T_m^v, \tag{11}$$

where N represents the neighbor particles count, with the rest of the rules repeating those for transitioning from bulk to spray ①. N was computed by counting the number of particles within a distance d_N up to a maximum of $N_{\max}$. We used defaults of $d_N = 5r_{\text{parent}}$ and $N_{\max} = 50$. We chose to use N rather than particle-per-voxel density n here because there is no voxel grid associated with the SPH spray particles.

Spray and mist particles transition back into bulk particles ④ when

$$\alpha > T_b^\alpha, \tag{12}$$

$$\phi < T_b^\phi. \tag{13}$$

Because the spray and mist particles entering the bulk water would be of various radii, and our bulk water assumes a uniform radius, we had to take additional steps to ensure this transition was momentum conserving. Upon transition, the velocity of mist and spray particles was transferred to the bulk water weighted by their mass. After the velocity transfer, noting that bulk water particles only act as markers in the volume, the radii of the newly transitioned particles were updated to be that of the rest of the bulk particles.

We also decided to allow bulk water particles to directly transition into mist ③. Our SPH solver experienced stability issues and required a higher number of substeps when spray particle radii became too small. When bulk particles were split into many smaller particles during transitions to spray, we immediately put the smallest particles under a radius threshold into mist, bypassing the spray state and thus avoiding this SPH instability.

4.2.3 *Air coupling.*

We simulated the air surrounding water as an Eulerian fluid, coupled together with each of the water states. The coupling implementation varied depending on the representation of the state. For spray and mist—Lagrangian particulate states—we used the coupling scheme from [Wretborn et al. 2022], via buoyancy and drag forces. Inspired by [Daviet and Bertails-Descoubes 2017], we generalized the drag force to an arbitrary non-linear degree c

$$F = \chi_a \mu_a r \left[6\pi + \frac{\pi}{4} \left(\frac{2r\rho_a \|\Delta \boldsymbol{v}\|}{\mu_a} \right)^{c-1} \right] \Delta \boldsymbol{v}, \tag{14}$$

and found the values $c \in [2.0, 2.4]$ to produce the most natural breakup patterns between particle sizes of different scales. The dependence on particle size in the drag force causes the trajectory of tiny mist particles to be dominated by the air, while large spray particles more easily fall to the ground. Here χ_a, μ_a, and ρ_a are dimensionless drag coefficient, dynamic viscosity, and density of air, respectively. r is the radius of the affected water particle, and $\Delta \boldsymbol{v}$ is its velocity relative to air. We found $\chi_a \in [2.0, 4.0]$ to work well in practice. Since water is much heavier than air we found that the buoyancy contribution did not have any significant effect, and discarded it for efficiency.

For air interacting with bulk water we initially explored a strongly coupled scheme via a single incompressible two-phase pressure solve. While this approach is accurate, it provided little artist control since there was no straightforward way to change how much water is affected by the wind, and vice versa. As a result we instead resorted to a weakly coupled scheme as follows: since air is much lighter than water, we used the signed distance field (SDF) of the bulk water as a collider for the air's incompressibility solve. In turn, the air affected bulk water via a drag force, applied explicitly to the surface layer of liquid particles at each timestep.

Using the water SDF as a kinematic collider with the air occasionally created small trapped air pockets that would get squeezed, resulting in sudden spikes in velocity. This led to instabilities when these regions of air would drag the water, and caused unnatural spurts of spray and mist. To avoid these issues, we clamped the relative velocity between the air and water to 10 m/s when evaluating all drag forces.

4.2.4 Discussion. The Loki state machine was crucial for delivering the volume of large-scale water shots in *Avatar: The Way of Water* while maintaining a consistent look, especially considering a large number of the shots were completed by junior artists. Typical large-scale water workflows require several separate iterations and supervisor approvals, first simulating the bulk water, then adding whitewater effects in layers as a post-process, and so forth. Our approach allowed us to get many of these layers completed and approved in a single pass. This also had the added benefit of providing a better "connectivity" between the states because they were coupled together in the same simulation.

One of the primary goals of this state machine approach was that it should generalize well across a wide variety of scenarios, scales, and resolutions. We were generally successful in achieving this goal. We also provided a "state machine lite" tool that would apply our transition and splitting techniques on pre-cached simulations where additional spray and mist elements were needed.

4.3 Anisotropic surfacing

We employed the approach of [Yu and Turk 2013] to create a renderable surface from simulated particles, which worked especially well for thin tendrils and sheets. However, we made a few modifications that we describe here. Following the original paper and using their notations we computed a singular value decomposition (SVD) of the covariance matrix $\boldsymbol{C}$ as

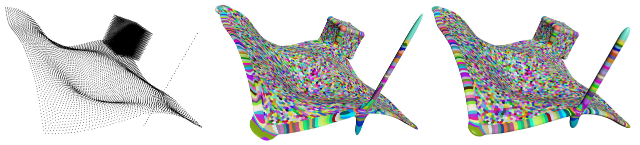
Figure 7: A test particle set with 1D, 2D, and 3D features (left); ellipsoids from [Yu and Turk 2013] implementation (middle) and from our approach (right). ©Wētā FX.

$$\boldsymbol{C} = \boldsymbol{R}\Sigma\boldsymbol{R}^T, \tag{15}$$

$$\Sigma = \text{diag}(\sigma_0, \sigma_1, \sigma_2), \tag{16}$$

where $\boldsymbol{R}$ is a rotation matrix with principal axes as column vectors, and Σ is a diagonal matrix with eigen values $\sigma_0 \geq \sigma_1 \geq \sigma_2 \geq 0$. Unlike the original paper, we then defined

$$\boldsymbol{G} = \sqrt{\Sigma^{-1}}\boldsymbol{R}^T, \tag{17}$$

$$\sqrt{\Sigma^{-1}} = \text{diag}(1/\sqrt{\sigma_0}, 1/\sqrt{\sigma_1}, 1/\sqrt{\sigma_2}), \tag{18}$$

and used it to build the anisotropic kernel

$$W(\boldsymbol{r}, \boldsymbol{G}) = P(\boldsymbol{G}\boldsymbol{r}), \tag{19}$$

where P is a kernel of a unit sphere. The need for the square root comes from an observation that σ_i are in fact the squares of the principal axes of the best-fitted ellipsoid to the input particles. This can also be seen from the dimensional analysis, as the units of $\boldsymbol{C}$ are $[m^2]$ by construction. We have found that this simple correction resulted in significantly smoother surfaces with fewer overly flat and spiky artifacts (see Figure 7). It also eliminated the need for clamping σ_i as proposed in the original paper. We only clamped all singular values to a fraction of the rasterization grid voxel size to avoid extremely thin ellipsoids that cannot be captured by the grid. Additionally, we found that the smoothing parameter $\lambda = 1$ gave the best visual results, which fits well with the core idea of the method.

In order to reduce aliasing artifacts when building the final representation of the surface, instead of building a density field by splatting and adding density functions W of ellipsoids, we splatted approximate SDFs of ellipsoids as deformed SDFs of unit spheres, and combined them via a minimum operation. As a result, we changed the meaning of P from a fog volume to an SDF of a unit sphere.

4.4 Thin film and drips

To make human-sized characters appear wet, with detailed sheets of water covering and rolling off of them as (can be seen in Figure 4), we utilized the approach of [Stomakhin et al. 2019]. A 1-5 cm voxel size bulk water simulation surrounding characters was enhanced by an additional bulk water pass at <1 mm voxel size—dubbed *thin film*—with viscosity, surface tension, and contact angle extrapolation to achieve adhesion and natural breakup into tendrils and droplets.

In order to be able to increase resolution this drastically and still have simulations complete in reasonable amounts of time we had to limit them to areas of visual significance. We decided to emit thin film particles from the contact line between a character and the surrounding bulk fluid surface and immediately cull them on reentry back into the bulk (see Figure 8). To achieve this practically, we scattered points over the character surface. At each timestep we checked which points exited the bulk water, and splatted them to make a FAB emitter volume, typically 1-3 voxels thick, depending on how much thin film needed to be pulled out of the bulk water for a specific shot. We set the emission velocity to the local velocity of the character mesh, since the motion of thin layers of water next to colliders is largely dominated by viscosity forces.

4.5 Adding details to surfaces

Residual wetness and rivulets. We created a new geometry-based residual wetness system implemented as a Houdini digital asset (HDA) that was capable of generating realistic droplets, trickling rivulets, and flowing rain water. This technique was used on a wide range of scenarios, including characters emerging from water, soaked in the rain, and sweating due to exertion (see Figure 9). Initially this system was written to create convincing static wet appearances for characters in shots following those with character-fluid interactions. As the show progressed we ended up using it in dynamic scenarios in conjunction with the FX simulations.

Our residual wetness system created approximations of two visual phenomena: first, static distributions of droplets that resulted from coalescing water, and second, dynamic water streams trickling in rivulets down a surface. Note that this "surface only" technique would not support dripping effects, so it naturally complemented

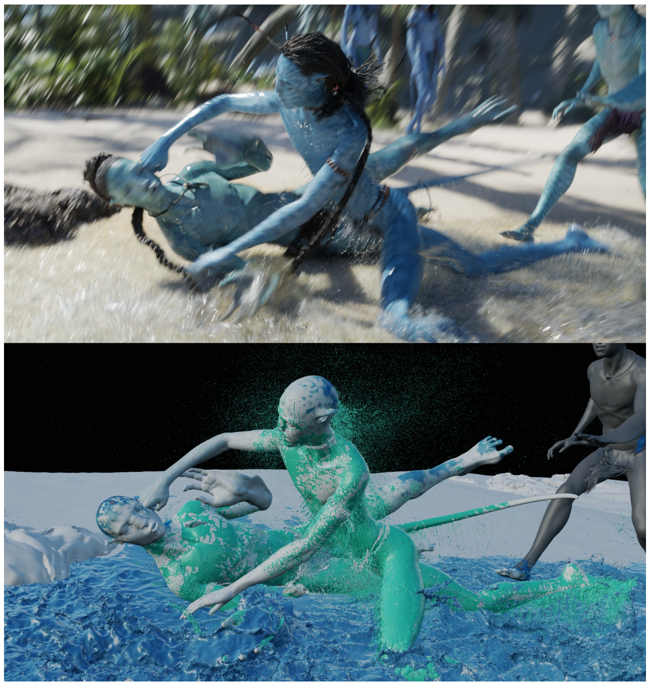
Figure 8: We employed the thin film approach of [Stomakhin et al. 2019] to run high resolution FLIP simulations (voxel size $\lesssim$1 mm) on characters' surfaces to achieve believable dripping effects. The bulk water surface (blue) was used to source and sink thin film particles (green) for efficiency, limiting the simulation to areas of visual significance. ©Disney and Wētā FX.

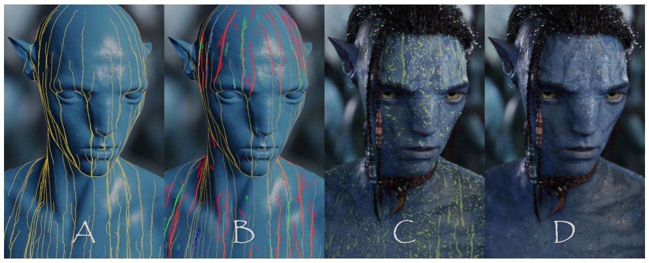

Figure 9: Different stages of our residual wetness approach: (a) guide curves, (b) procedural rivulet animation, (c) meshed rivulets and droplets, and (d) the final frame. ©Disney and Wētā FX.

our thin film solution described previously, as the latter had a harder time keeping water droplets stuck to the surfaces or enforcing the flow to adhere to the prescribed stream patterns.

We generated the distribution, shape, and movement of the output in a deterministic manner. This allowed for an independent evaluation of individual frames without relying on a full sequential simulation. The HDA was run directly within Katana during expansion to the renderer, via HoudiniEngine. We picked a reference frame from a shot, typically defaulting to the first frame, to use when generating the patterns and curves, resulting in a contextual registration of the action to the character.

We achieved the appearance of static coalesced surfaces by advecting an initially uniform distribution of points. We provided 120 instanced droplet exemplars to the renderer, which it often instanced up to 10,000 times on a single character.

We created plausible dynamic rivulet patterns by first recursively generating multiple octaves of guide curves over the surface. For each octave we started with a distribution of initial points on the surface. For each of those points, we walked down the surface, creating a curve attracted to any previously created octaves within its proximity. This attraction mimicked the behaviour of real-world rivulets, which tend to favor existing paths of water when running down a surface. We used textures from the surface to control where the rivulet paths started, and where the paths could travel. This provided us with artistic controls to fit the appearance to the required scenarios within shots. In shots requiring thin film simulations, we used a thin film VDB surface from Loki to provide masking for the reveal of the residual wetness appearance. Once we generated a complete rivulet path, we assigned it a probability of moving, and a random starting time. Given these variables we could determine how far along the path the head of the rivulet was for the requested frame. For each point on the path, we calculated the distance to the head and used this as a lookup for the height of the trail in a user-defined spline, with a maximum value of 0.4 cm.

To achieve a convincing contact angle, we tucked each point below the surface by a distance of 0.4 cm minus the height. To create a desired width of the rivulet, we duplicated the points and translated them perpendicular to the curve direction. We then surfaced the paths with Houdini's spherical particle surfacing method, using a point radius of 0.4 cm. To better integrate the resulting shapes, we added the high-detail displacement from the underlying geometry to the resulting mesh, when the height of the trail was low.

This method offers several improvements over previous textural and shader-based techniques:

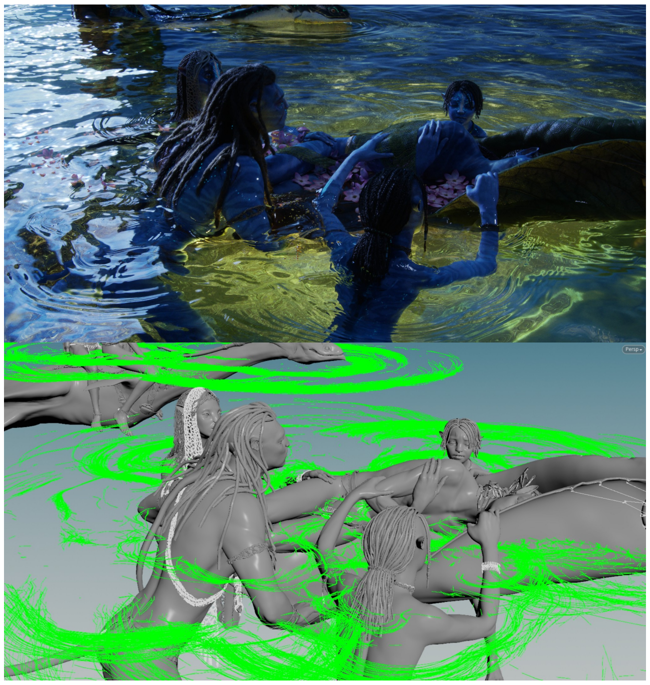

Figure 10: Using the wave curve approach of [Skrivan et al. 2020] to create tiny surface-tension-driven ripples on top of our bulk fluid surfaces. ©Disney and Wētā FX.

- Renders accurate caustics and shadows within and beyond the droplets. This was particularly apparent when the droplets would sit on a surface grazing with respect to the light.
- Preserves the appearance of the underlying materials.
- Avoids appearance filtering by generating rivulet mesh independent of the camera, unlike texture-based approaches.

Capillary waves. Our bulk fluid solver captured details only at scales larger than its voxel size, which would rarely go below 1 cm for typical water tank scenarios. To restore the missing detail, we utilized the approach of [Skrivan et al. 2020], which allows adding tiny surface-tension-driven ripples to simulated water surfaces as a post-process. We found this method to work well out of the box with default settings (see Figure 10).

We discovered that in relatively calm water scenarios we could bypass bulk water simulation altogether, and run the capillary wave solver directly on top of the procedural ocean waves. In order to do so we only needed to create a velocity field for wave curve emission. We built the velocity field at each frame by taking the velocity field of the procedural water waves and performing a divergence-free projection on it in the presence of the collisions geometry. This gave us plausible-looking wave curve distributions generated naturally off of the colliders.

4.6 Secondaries

For bubbles and foam we used the approach of [Wretborn et al. 2022], which allows for two-way coupling between so called *diffuse bubble* particles and the surrounding bulk water, with subsequent seamless transition to SPH foam. The most common use case for this method was to run it as a secondary simulation, re-simulating a local region of the bulk water to guide the underwater bubbles, and using the bulk water surface as an advection manifold for the foam. For a few select shots where the bubbles' influence on the bulk water could not be ignored, such as the one shown in Figure 16, we simulated bubbles and foam simultaneously with the other water states, as indicated in Figure 6. The only required change for this was to couple bubbles directly with the bulk water instead of a sparsely allocated guided fluid.

Artist time spent doing secondary simulations often came down to two processes: tuning emission criteria, and preparing inputs—such as fluid surfaces and colliders—for simulation. Our focus lay in finding standardized methods for these processes that could be transferable to many different scenarios.

We addressed the first part—emission—with the aeration metric and volume-based emission strategy proposed in [Wretborn et al. 2022], which creates physically plausible emission patterns from bulk fluid simulations (⑤ in Figure 6). This covered the majority of shots that required bubbles and foam secondaries. For areas outside of the bulk fluid region, where aeration would not be available, artists resorted to painting custom emission-density maps.

4.6.1 Preparing input. The main input that needed to be prepared was the water SDF. For the vast majority of shots only a shallow layer of bulk water was simulated, which would often not extend to the depth to which bubbles can realistically travel. To avoid bubbles constrained to the inside of the original bulk-water simulation domain, artists would often extend its SDF by iteratively transforming it downwards and combining with itself. This process was cumbersome and difficult, especially for simulation domains with complex geometry. What is more, the domain cannot be expanded horizontally this way, which severely limits the potential foam regions. This was especially challenging for shots with trailing foam, such as the one seen in Figure 11.

We solved this problem by using a composite representation of the water surface. Seamless blending of the simulation water region into the surrounding procedural ocean waves was already

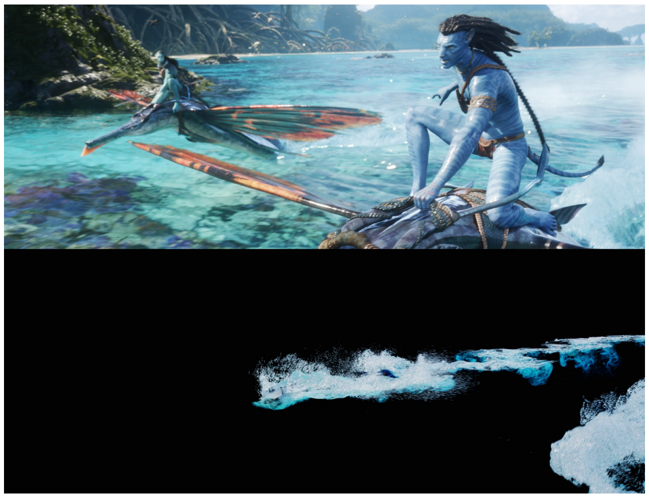

Figure 11: Long trails of bubbles (blue) and foam (white), simulated using our composite water surface representation. ©Disney and Wētā FX.

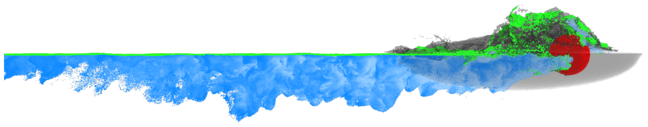

Figure 12: Running a bubble and foam simulation on an implicit composite surface. The bulk water SDF is shown as a transparent surface (gray) around the moving object (red). As foam (green) and bubbles (blue) reach the edge of the SDF they seamlessly transition to being simulated using the implicit procedural open-ocean surface. ©Wētā FX.

an important part of our delivery from FX to rendering, which was accomplished using Loki's render graphs, such as the one shown in Figure 3. We were able to utilize the output of the same compositing graph for our secondaries simulations. The composite fluid surface extends infinitely far in all directions, imposing no limitation on the simulation region for bubbles or foam, and can be dynamically sampled at any point in space, removing the burden of artists to define the simulation region upfront. An example of this setup can be seen in Figure 12.

4.6.2 Secondaries at different scales. A common challenge for whitewater systems is the ability to simulate scenarios on a wide variety of scales. [Wretborn et al. 2022] established a set of dynamic parameters that produce compelling results for close-up scenarios, where you can computationally afford realistic bubble radii r in $[1, 10]$ mm range and voxel sizes of a few centimeters. As the camera gets further away from bubble and foam action, it makes sense from both rendering—no need to capture detail below screen pixel size—and simulation cost standpoints to increase bubble particle sizes. Each of the bigger particles can then be interpreted to represent an aggregate of actual bubbles. As an aggregate has different dynamics from an individual bubble of the same size, the parameters of the original model needed to be readjusted. In what follows we describe the details of that adjustment.

Particle size and guided volume voxel size. Since bubble particles are the primary objects an artist sees, we devised a workflow where they could specify the *smallest feature size* L viewable in the scene. They would then have the size of particles and voxel size set proportional to the unitless *length scale* $l = L/1$ mm, relative to what those sizes would have been in the close-up setting of [Wretborn et al. 2022]. Our simulations would typically have $l \in [1, 15]$.

Adjusting drag. The drag model of [Wretborn et al. 2022] was designed for small individual bubbles submerged in a fluid; however it is not valid for larger aggregates of bubbles. Assume a particle with radius R represents an aggregate of $N_R = (R/r)^3$ smaller bubbles with radius r, where $R = lr$, and for simplicity assume the drag force is linear with respect to bubble radii $F(r) \propto r$. The underestimation of the drag force on the aggregate particle is then proportional to

$$\frac{N_R F(r)}{F(R)} = \frac{N_R r}{R} = \left(\frac{R}{r}\right)^2 = l^2, \tag{20}$$

since it is in fact composed of N_R smaller ones. We compensate for the difference by adjusting the aggregate particle drag coefficient by a factor of l^2.

Adjusting cohesion and pressure. A similar argument to that of the above was attempted for cohesion, however the resulting scaling removed a lot of the interesting foam structures and looked unsatisfactory. We found that as the particle sizes increased so did their relative acceleration, due to cohesion, for the same SPH settings. Thus, instead of correcting the cohesion strength, we empirically found that increasing SPH pressure stiffness linearly with l produced plausible results at larger scales.

4.7 Render delivery

We used Loki graphs, referred to as *render graphs*, to package simulation outputs for rendering which could be used for visualization, final renders within Manuka, or general sampling for further simulation refinement. These render graphs are able to sample each Pahi component at the original resolution, then blend them together, with discrete inputs such as simulations, continuous inputs such as open-ocean waves, and hybrid representations such as wave curve displacement (see Figure 3). For a volume Loki builds and evaluates a render graph as one would a shading graph per volume sample. For example, sampling a VDB cache at a given position, combining two samples into one based on their value and a mask, or projecting the sample position onto an implicit surface. Deferring evaluation of render graphs let Manuka adaptively choose where to sample the water, and then per-sample load and blend all the required simulation components, thereby avoiding degradation of intermediate discretizations into common domains. Render graphs also allowed both FX and lighting artists to make updates post-simulation in order to fine tune each water element—or even evaluate it outside of a rendering environment—such as passing it back to animation as a visualization guide for additional animation passes.

Similar to the simulation regimes, there was no single rendering primitive which efficiently captured all the water visual requirements from large-scale oceans to small scale rivulets on characters or misty splashes. We mixed and matched a suite of water primitives depending on the outputs of the simulations and the needs of the shot. Some water primitives represented solid water, or thin-wall features such as bubbles, or even regions of heavily mixed air and water such as foam or aeration, all requiring careful consideration of what the behavior should be above and below an existing water line, as seen in Figure 13.

Implicit Surfaces. The most general and also most widely used render primitive was an SDF delivered to Manuka through the ImplicitField RenderMan API. We sampled the implicit surfaces using a custom Manuka procedural which evaluates a Loki render graph per volume sample, to allow for blending of multiple inputs without discretizing into a common domain. Each volume sample consists of a sample position and a filter size, to avoid aliasing when sampling more coarsely than the underlying data. We largely generated the individual source inputs into the SDF render graph before render time. They consisted of one of:

1. **Discretized level sets.** Inputs such as bulk simulation particles, surfaced into an SDF and cached out as a VDB.
2. **Fully procedural inputs.** Inputs such as open-ocean wave height fields, created at render time through mathematical transformations of the user parameters into an SDF.

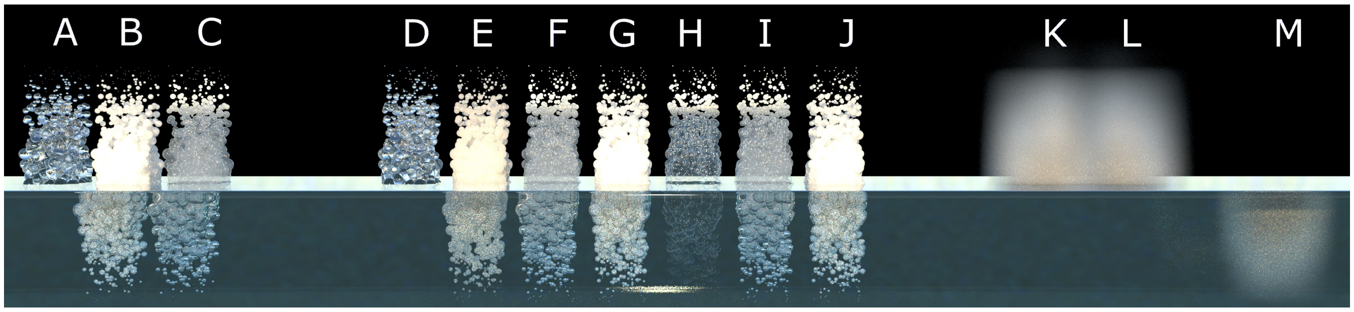

Figure 13: The most common water rendering primitive and material combinations shown above and below the water line. The first three from the left are all using an implicit surface rendering primitive with a spray (A), foam (B), and bubble (C) material. The next group in the middle is rendered with particle primitives and can switch to isotrace on demand. The materials are spray (D), spray diffuse (E), foam (F), foam diffuse (G), foam thinwall (H), bubbles (I), and bubbles diffuse (J). The final three are rendered with a volumetric rendering primitive and a spray (K), foam (L), and aeration (M) material. ©Wētā FX.

(3) **Hybrid discretized procedural inputs.** Inputs such as wave curves and volume instancing, using a mixture of analytic components and pre-cached placements to generate local signed distance fields.

We then blended together the individual inputs with compositing operations in the render graph, by sampling each of them and combining with constructive solid geometry (CSG) operators such as union. Sometimes masks were required to guide where to apply the blend, for example to limit the blend between a simulated splash and a procedural open-ocean SDF, to ensure the splash impact would not be clipped by the ocean surface. The Loki render graph communicates additional information to Manuka, such as the bounds of the data, and the minimum feature size for a given area in order to guide Manuka on where to sample, while still leaving Manuka to make the final decision on sample locations and filter sizes.

Particles. The next most common render primitive was particles represented as RiPoints through the RenderMan API and delivered through Loki compositing graphs. We used them for representing small bubbles or water spray small enough to not need complex geometric detail, but sparse and distinct enough that a volumetric representation would be too diffuse. We cached out most particles from FX simulations and then passed them through a Loki compositing graph at runtime in order to provide last-minute control, such as culling by density, radius adjustment, or projecting to the water surface.

Isotrace Primitives. The isotrace plugin to Manuka extends particles with additional attributes to represent several shapes analytically and provides efficient ways to directly trace against them. The blobby isotrace shape was the most used and provided a way to deliver particles without fully surfacing them while still representing a smooth connectivity between them, using the approach from [Sabbadin and Droske 2021]. Blobbies were especially useful for the intermediate spray state, separating from the bulk water but still retaining some distinct shape before eventually becoming so sparse that more simple small particles or a volumetric representation would suffice.

Volumes. We also delivered density-based volumes through the ImplicitField RenderMan API via a Loki render volume compositing graph, often used to represent very aerated regions of water or regions of fine mist.

Meshes. In limited cases, we used meshes when there was no need for blending with other elements, while still requiring more complex geometry than was available with particles or isotrace primitives. These cases were mostly procedural and included the residual wetness and rivulets attached to characters, or marine snow to represent small floating bits of debris in the water.

5 PRODUCTION USE

During *Avatar: The Way of Water* production, the FX team was tasked with bringing the magical water world of Pandora to life. From complex reef villages to enormous characters leaping out of the water, it had to be something the audience had never seen before, yet also familar and grounded in physics. We will outline some of of our workflows and production example cases below.

5.1 Kaita template

We utilized a proprietary Houdini template manager called Kaita that allowed artists to load templates which included all the tools and solvers required to run a given shot.

Kaita water templates included a wide range of tools/HDAs from asset loading, procedural ocean spectra, simulation preparation, bulk water simulation, bubble/foam secondary simulation, thin film simulation to publishing and rendering, following the data flow outlined in Figure 2.

Within these templates artists would have custom presets, such as the thin film template where the highest quality used in production was a voxel size of 0.6 mm and 20 substeps pre frame, and the lowest quality was a voxel size of 0.1 cm and 10 substeps per frame. For Loki state machine simulations the standard delivery resolution was a voxel size of 5 cm with 5 substeps per frame, giving great resolution and detail, while also keeping the simulation times reasonable.

Utilizing the Kaita template structure enabled artists to work through a large number of shots of varying complexity, while also ensuring consistency. For efficiency there was a predefined structure to the templates, which enabled senior artists, leads, and supervisors to efficiently triage and debug scenes without having to decipher custom scene files or unique workflows per artist. There

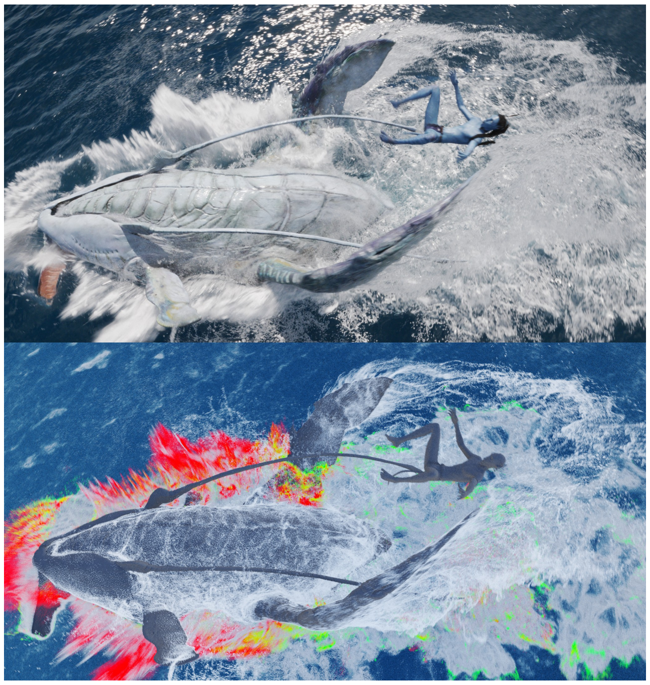

Figure 14: This shot of Payakan and Lo'ak playing together in the water was completed in a single simulation pass using Loki state machine for bulk water (blue), spray (green), mist (red) and air (not visualized). The bubbles and foam, and thin film simulations were performed in separate passes. ©Disney and Wētā FX.

were weekly development round tables where any innovation or improvement could be discussed, consumed, and deployed into the templates for wider use.

5.2 Large scale splashes

One of the challenges the Simulation and FX teams faced during production was water at large scales. Lo'ak meeting the 28 m tulkun Payakan is a great example that presented significant challenges (see Figure 14). With Payakan being roughly the size of a blue whale, getting believable interaction and splashes at this scale was paramount. We approached this as a single state machine solve, with fully coupled water, spray, and mist; for thin film, bubbles, and foam we used directional coupling. "Directional coupling" is what we refer to when reusing physics data from the primary solve, such as velocity fields, air solves, or anything useful in additional solves. By using this technique we were able to integrate these varying elements into the same physical world, where large splashes, spray, and mist react with the main body of water, producing aeration, bubbles, and foam.

Where required we also used directional coupling to provide additional "sweetener" passes, allowing us to hit specific creative notes from the director. Taking the solved underlying bulk water, we could run a secondary solve, with some slight modifications to transition behaviors, such as how quickly we wanted to transition into spray and/or mist, or even transition straight from bulk water to mist if it met certain criteria. The simulation time for the final primary solve was 18 hours on a single 512 GB machine, and 6 hours per secondary directionally coupled element, at a voxel resolution of 5 cm and 2 substeps per frame.

While the Loki state machine had already been used on numerous shots of varying scales at this stage in the project, we had not yet approached something this large. Once we found the right settings, we could deploy this recipe on other Payakan jumping shots or shots of a similar scale. Within this framework, we needed relatively few iterations per shot before showing to the client for creative approval.

To achieve the final rendered look, we surfaced the bulk water with our anisotropic surfacer with *aeration roughening*: a technique to thin out areas where transition occurs by scaling down the sizes of splatted ellipsoids. The ballistic deliverables were blobbies (spray), points (particulate mist), and volumes (volumetric mist). A balanced combination of all these deliverables allowed us to achieve the smooth transition through the water states.

5.3 Underwater bubbles

We were tasked with creating numerous above and below water scenarios, and in some cases, a mixture of both. For underwater shots or shots at the waterline, the entrainment of air in water was a key component to selling realisim. A typical recipe for underwater shots would usually include aeration volumes to create the smallest diffuse bubbles (discussed in Section 4.6), and high-fidelity *hero bubbles*: a term we used to describe large volumes of air trapped under water with explosive dynamics, where capturing the evolution of the air-water interface was essential. We followed the approach of [Stomakhin et al. 2020] to simulate the latter (see Figure 15). In some scenarios hero bubbles were required to break the surface and pop. For these shots, we simulated the hero bubbles first, and then used them as guide velocity fields or collision inputs for the subsequent bulk-water simulation.

[Stomakhin et al. 2020] shows that it is possible to have hero and diffuse bubbles coexist with a single simulation. However, we found this approach difficult to use in practice since similar sized hero and diffuse bubbles can have different apparent dynamics. As a result, for scenarios where both diffuse and hero bubbles were present (see Figure 16), we simulated the representations separately.

To avoid rendering diffuse bubbles as perfect spheres, we followed [Patkar et al. 2013] and instanced evolving shapes onto the diffuse bubble particles. For hero bubbles we surfaced the air FLIP

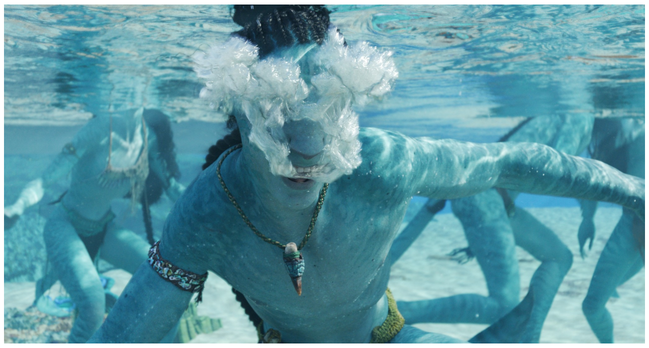

Figure 15: The release of large volumes of air under water, simulated using the hero bubble approach of [Stomakhin et al. 2020]. ©Disney.

particles with our anisotropic surfacer and rendered the resulting SDF.

6 LESSONS AND LIMITATIONS

When developing a large water system, one inevitably has to make difficult design decisions and compromises, as different production requirements are at odds with one another, such as art-directability versus physicality. With a development cycle of multiple years, like that of the massive VFX production of *Avatar: The Way of Water*, it is often hard to predict how these choices will hold up in the long run. We now summarize the compromises we made when designing Pahi, and discuss how those have worked out in practice.

Physics vs artistic control. Despite longer simulation times, we chose physics-based workflows to help reduce artist iterations. This helped achieve the most consistent look and quality of the results across a large number of TDs. This came at a cost of willfully constraining ourselves to less flexible or art-directable setups, compared to other systems, such as SideFX Houdini by default. For example, several heavily art-directed shots required senior artists to stray away from the primary workflow to do one-off setups. But we found the overall trade-off a major win.

Coupling vs directional simulation. Accurate coupling, such as bubbles in water or airborne spray in air, allowed us to achieve unprecedented visual results. At the same time, having sequential simulations with directional interaction was important for faster turnaround times with the ability to address notes on individual components. For example, we sometimes ran foam and bubbles as a secondary simulation, even though our system was capable of full coupling with the rest of the state machine. We found it valuable to have multiple coupling approaches at hand; this gave the artists the flexibility to choose the most optimal method depending on the requirements of the shot.

Single vs multi-state representation. A single water representation seems desirable from a physical-accuracy and reduced-management-complexity standpoint. However, that would be far from practical computationally. On the other hand, separate representations for different water states can be optimized accordingly, but then the interaction between them becomes a challenge both from simulation (coupling and transitions) and rendering (different scales and shading) perspectives. Our solution was a compromise, where we managed a few distinct water states, but tried to not have unnecessary ones. We used the Loki state machine to handle the states we chose, ensuring seamless momentum-conserving transitions. We applied a similar approach to rendering deliverables, using a variety of primitives to best fit each specific use case rather than one approach for all types of water. The Loki render graphs allowed for drastic differences in the simulation resolutions and a consistent system to manage the mapping of simulation outputs to rendering primitives.

7 CONCLUSION

Avatar: The Way of Water's ambitious development pushed the state of the art for many aspects of water. Rather than prescribing a one-size-fits-all solution, a new cross-department workflow facilitated development, experimentation, and artist flexibility while ensuring the large suite of components seamlessly integrated back into a cohesive whole.

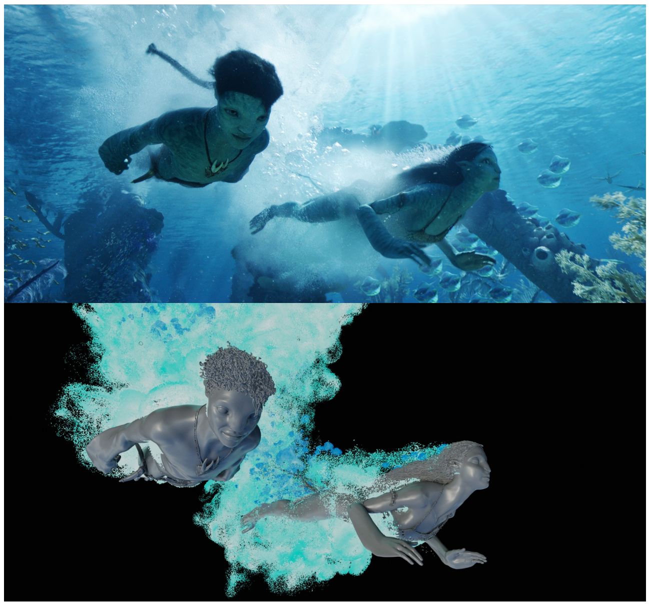
Figure 16: This shot of characters diving into water was completed using a combination of diffuse (cyan) and hero (blue) bubbles each coupled with the surrounding water. ©Disney and Wētā FX.

ACKNOWLEDGMENTS

We are thankful to the many people besides the authors of this article who contributed to Pahi and the varied water components. We thank the Simulation Research department, including John Edholm, Noh-hoon Lee, Nikolay Ilinov, and Marcus Schoo who developed major pieces of Loki fluid simulation, surfacing, and render delivery functionality. Eston Schweickart, Xiao Zhai, and Gilles Daviet brought fluid and elastic dynamics together for wet character simulations. Andrew Moffat, Job Guidos, and Tim Ebling bridged simulation research and FX production. Andrew Sidwell, Martin Loga, and Christopher Horvath developed much of the procedural open ocean wave deformer library. Tomas Skrivan and Christoph Sprenger provided wave curve and shallow water solvers. Marc Droske, Manuele Sabbadin, Tomas Davidovic, Andrea Weidlich advanced the fundamental rendering of water.

We thank the FX department for the varied artistic, technical, and leadership contributions. Gary Boyle designed much of the original Pahi framework and API. Chet Leavai was key in early water development and implicit rendering workflows. Christos Parliaros pushed the animation FX and creature fluid coupling workflows. Katie Talbot, Steffanie Blatt, Vitto Maglione, and Ziggy Kucas led major Loki solver testing during development. James Robinson, Robert Wilkins, Tanē MacDonald, Shreyas Sood, Jackson Preston, Brock Trewavas, Lucy Jaegers, Sandy Sutherland, and Alexander Jarosch contributed to Pahi during production. Thank you to Alex Nowotny, David Caeiro Cebrian and David Morton, Eddy Purnomo, Boris Bruchhaus, Brandon Fleet, Ryan Bowden, Anthony Arnoux for their FX Leadership for water simulations in production.

Joe Churchill, Guillaume Francois, and Chris George led major water lookdev work. John Homer, Andrea Merlo, Carlos Lin, Matt Muntean, and Tim Hawker adopted fluid simulations into their creatures workflows. Alex Klaricich developed much of the lighting workflow managing all the varied water deliveries. Louis-Daniel Poulin was instrumental getting the residual wetness system off the ground. Sam Cole's leadership in compositing smoothly brought together thousands of people's work.

Thank you to James Cameron and Jon Landau whose vision pushed forward the entire VFX industry, especially around water. Many of the visual effects supervisors, especially Pavani Rao Boddapati, Eric Saindon, Nick Epstein, and Chris White, provided multiple years of invaluable feedback. And finally, thank you to Joe Letteri who gave us the time, resources, and tireless guidance to deliver a more holistic and physically principled production water system.

REFERENCES

Christopher Batty and Robert Bridson. 2008. Accurate Viscous Free Surfaces for Buckling, Coiling, and Rotating Liquids. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Dublin, Ireland) *(SCA '08)*. Eurographics Association, Goslar, DEU, 219–228.

Robert Bridson. 2015. *Fluid simulation for computer graphics.* CRC press.

G. Daviet and F. Bertails-Descoubes. 2017. Simulation of Drucker–Prager granular flows inside Newtonian fluids. (Feb. 2017). Working paper or preprint.

Luc Deike, W Kendall Melville, and Stéphane Popinet. 2016. Air entrainment and bubble statistics in breaking waves. *J. Fluid Mech.* 801 (Aug. 2016), 91–129.

Luca Fascione, Johannes Hanika, Mark Leone, Marc Droske, Jorge Schwarzhaupt, Tomáš Davidovič, Andrea Weidlich, and Johannes Meng. 2018. Manuka: A Batch-Shading Architecture for Spectral Path Tracing in Movie Production. *ACM Trans. Graph.* 37, 3, Article 31 (aug 2018), 18 pages. https://doi.org/10.1145/3182161

Florian Ferstl, Ryoichi Ando, Chris Wojtan, Rüdiger Westermann, and Nils Thuerey. 2016. Narrow Band FLIP for Liquid Simulations. In *Proceedings of the 37th Annual Conference of the European Association for Computer Graphics* (Lisbon, Portugal) *(EG '16)*. Eurographics Association, Goslar, DEU, 225–232.

Chuyuan Fu, Qi Guo, Theodore Gast, Chenfanfu Jiang, and Joseph Teran. 2017. A Polynomial Particle-in-Cell Method. *ACM Trans. Graph.* 36, 6, Article 222 (nov 2017), 12 pages. https://doi.org/10.1145/3130800.3130878

Jonathan Garcia, Sara Drakeley, Sean Palmer, Erin Ramos, David Hutchins, Ralf Habel, and Alexey Stomakhin. 2016. Rigging the Oceans of Disney's "Moana". In *SIGGRAPH ASIA 2016 Technical Briefs* (Macau) *(SA '16)*. Association for Computing Machinery, New York, NY, USA, Article 30, 4 pages. https://doi.org/10.1145/3005358.3005379

Christopher J. Horvath. 2015. Empirical Directional Wave Spectra for Computer Graphics *(DigiPro '15)*. Association for Computing Machinery, New York, NY, USA, 29–39. https://doi.org/10.1145/2791261.2791267

Richard Jones and Richard Southern. 2017. Physically-Based Droplet Interaction *(SCA '17)*. Association for Computing Machinery, New York, NY, USA, Article 5, 10 pages. https://doi.org/10.1145/3099564.3099573

Steve Lesser, Alexey Stomakhin, Gilles Daviet, Joel Wretborn, John Edholm, Noh-Hoon Lee, Eston Schweickart, Xiao Zhai, Sean Flynn, and Andrew Moffat. 2022. Loki: A Unified Multiphysics Simulation Framework for Production. *ACM Trans. Graph.* 41, 4, Article 50 (jul 2022), 20 pages. https://doi.org/10.1145/3528223.3530058

Ken Museth. 2013. VDB: High-Resolution Sparse Volumes with Dynamic Topology. *ACM Trans. Graph.* 32, 3, Article 27 (jul 2013), 22 pages. https://doi.org/10.1145/2487228.2487235

Sean Palmer, Jonathan Garcia, Sara Drakeley, Patrick Kelly, and Ralf Habel. 2017. The Ocean and Water Pipeline of Disney's Moana. In *ACM SIGGRAPH 2017 Talks* (Los Angeles, California) *(SIGGRAPH '17)*. Association for Computing Machinery, New York, NY, USA, Article 29, 2 pages. https://doi.org/10.1145/3084363.3085067

Saket Patkar, Mridul Aanjaneya, Dmitriy Karpman, and Ronald Fedkiw. 2013. A hybrid Lagrangian-Eulerian formulation for bubble generation and dynamics. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Anaheim, California) *(SCA '13)*. Association for Computing Machinery, New York, NY, USA, 105–114.

Manuele Sabbadin and Marc Droske. 2021. *Ray Tracing of Blobbies.* Apress, Berkeley, CA, 551–568. https://doi.org/10.1007/978-1-4842-7185-8_35

Tomas Skrivan, Andreas Soderstrom, John Johansson, Christoph Sprenger, Ken Museth, and Chris Wojtan. 2020. Wave Curves: Simulating Lagrangian Water Waves on Dynamically Deforming Surfaces. *ACM Trans. Graph.* 39, 4, Article 65 (aug 2020), 12 pages. https://doi.org/10.1145/3386569.3392466

Alexey Stomakhin, Andrew Moffat, and Gary Boyle. 2019. A Practical Guide to Thin Film and Drips Simulation. In *ACM SIGGRAPH 2019 Talks* (Los Angeles, California) *(SIGGRAPH '19)*. Association for Computing Machinery, New York, NY, USA, Article 72, 2 pages. https://doi.org/10.1145/3306307.3328141

Alexey Stomakhin and Andrew Selle. 2017. Fluxed Animated Boundary Method. *ACM Trans. Graph.* 36, 4, Article 68 (jul 2017), 8 pages. https://doi.org/10.1145/3072959.3073597

Alexey Stomakhin, Joel Wretborn, Kevin Blom, and Gilles Daviet. 2020. Underwater Bubbles and Coupling. In *ACM SIGGRAPH 2020 Talks* (Virtual Event, USA) *(SIGGRAPH '20)*. Association for Computing Machinery, New York, NY, USA, Article 2, 2 pages. https://doi.org/10.1145/3388767.3407390

Joel Wretborn, Sean Flynn, and Alexey Stomakhin. 2022. Guided Bubbles and Wet Foam for Realistic Whitewater Simulation. *ACM Trans. Graph.* 41, 4, Article 117 (jul 2022), 16 pages. https://doi.org/10.1145/3528223.3530059

Jihun Yu and Greg Turk. 2013. Reconstructing Surfaces of Particle-Based Fluids Using Anisotropic Kernels. *ACM Trans. Graph.* 32, 1, Article 5 (feb 2013), 12 pages. https://doi.org/10.1145/2421636.2421641

A SPLITTING REPARAMETERIZATION

The average volume of a particle produced by sampling the radius distribution (1) is

$$\overline{V} = \int_{r_{\min}}^{r_{\max}} \frac{4}{3}\pi r^3 \Pi(r) dr = \frac{A}{4-\eta}\left(r_{\max}^{4-\eta} - r_{\min}^{4-\eta}\right)\frac{4}{3}\pi, \qquad (21)$$

where the normalization constant A can be found from

$$\frac{1}{A} = \int_{r_{\min}}^{r_{\max}} \frac{1}{r^\eta} dr = \frac{1}{1-\eta}\left(r_{\max}^{1-\eta} - r_{\min}^{1-\eta}\right). \qquad (22)$$

If k is the total number of emitted split particles, their total expected volume is $k\overline{V}$, which we would like to be equal to the volume of the parent particle $V_{\text{parent}} = \frac{4}{3}\pi r_{\text{parent}}^3$. From that

$$\frac{V_{\text{parent}}}{k} = \frac{4}{3}\pi \frac{1-\eta}{4-\eta} \frac{r_{\max}^{4-\eta} - r_{\min}^{4-\eta}}{r_{\max}^{1-\eta} - r_{\min}^{1-\eta}}, \qquad (23)$$

and substituting $r_{\max} = \xi r_{\min}$ we get

$$r_{\min}^3 = \frac{r_{\text{parent}}^3}{k} \frac{4-\eta}{1-\eta} \frac{\xi^{1-\eta} - 1}{\xi^{4-\eta} - 1}. \qquad (24)$$