Fluxed Animated Boundary Method

ALEXEY STOMAKHIN and ANDREW SELLE*, Walt Disney Animation Studios

We present a novel approach to guiding physically based particle simulations using boundary conditions. Unlike commonly used ad hoc particle techniques for adding and removing the material from a simulation, our approach is principled by utilizing the concept of volumetric flux. Artists are provided with a simple yet powerful primitive called a *fluxed animated boundary (FAB)*, allowing them to specify a control shape and a material flow field. The system takes care of enforcing the corresponding boundary conditions and necessary particle reseeding. We show how FABs can be used artistically or physically. Finally, we demonstrate production examples that show the efficacy of our method.

CCS Concepts: • **Computing methodologies** → **Physical simulation**; *Animation*;

Additional Key Words and Phrases: physical simulation, control, art-direction, FLIP, APIC, MPM

ACM Reference format:
Alexey Stomakhin and Andrew Selle*. 2017. Fluxed Animated Boundary Method. *ACM Trans. Graph.* 36, 4, Article 68 (July 2017), 8 pages.
DOI: http://dx.doi.org/10.1145/3072959.3073597

1 INTRODUCTION

Natural phenomena are compelling, important, and pervasive throughout computer graphics. While physical simulation is guaranteed to produce a plausible and realistic simulation, artists are paradoxically forced to continually re-run simulations to target story and director needs—the process of art-direction. Developer and artist time spent on art-direction far exceeds the effort required for core simulation technology.

Art-direction is achieved either by *internal forces and constraints* or by *boundary conditions*. The former is useful and direct but when overused undermines scene realism. The latter prescribes the connection between simulated material and the wider world—abstracting far-field details into the minimal information needed for simulation. Moreover, it will always be impossible and undesirable to simulate the whole world, so the rest of the world must be kinematically described or described with a lower resolution simulation. The most common boundary condition controls are kinematic solids, sources, and sinks, and these tend to create more realistic, naturalistic simulations.

Volumetrically defined boundary conditions were used frequently for Eulerian (grid) solvers, but have been downplayed in more recent Lagrangian (particle) solvers. FLIP/APIC and SPH solvers advocate simple creation and deletion of particles, which makes life seemingly simpler for an artist—a major source of the solvers' popularity. Any artist that knows how to use ubiquitous particle tools [Reeves 1983] can create and destroy fluid. Accurate sinks and sources, however, are not correctly modeled without the concept of volumetric flux. The effect is that artists using stock tools often embrace simple particle methods and live with poor boundary conditions. Even worse, practitioners tend to consider boundary conditions only of discrete objects like "the water" or "the collision object". This misses the possibility of amorphous boundaries that change the active subset of material being simulated. In particular, the view fails to help define the connection between simulated fluid and non-simulated fluid in an open ocean example (Figure 3b).

To solve this volumetric control problem for particle solvers, we introduce the *fluxed animated boundary (FAB)* method. An artist provides an animated shape and a material velocity field. The animated

Fig. 1. Examples of fluxed animated boundary (FAB) method used for open water simulations in Disney's Moana. Each image contains a simulated boat wake and the top image additionally features a fully simulated breaking barrel. ©Disney

* Andrew Selle is currently affiliated with Google.

0730-0301/2017/7-ART68 $15.00
DOI: http://dx.doi.org/10.1145/3072959.3073597

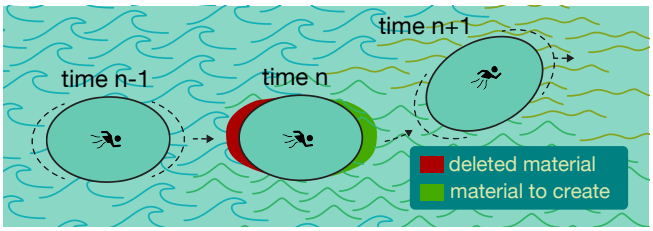

Fig. 2. A schematic illustration of a windowed simulation. An artist specifies a window shape that bounds an active simulated subset of the material local to a character. The material velocity field is specified on the outside, corresponding to the desired prescribed motion of the material. The goal is to make the transition of the material between being simulated and not as seamless as possible, regardless of the motion of the window. ©Disney

shape describes how the window bounding the active simulated subset changes over time, and the material velocity describes the prescribed motion of the material on the outside, see Figure 2. Given these primitives, the method integrates with standard FLIP, APIC or MPM solvers, see [Jiang et al. 2015], while automatically handling particle reseeding. Artists used this method on Disney's animated film Moana, finding it massively more intuitive and higher quality than particle-based control.

Our contributions are:

- An effective artist friendly volumetric control.
- Accurate particle reseeding in FLIP/APIC/MPM.
- A family of physical FABs including Stokes waves and deep water waves.

2 PREVIOUS WORK

3D liquid simulation was introduced to graphics by [Foster and Metaxas 1997], who also introduced a simple control mechanism. As fluid simulation became popular, fluid control was a common focus. Researchers showed how to match predefined shape keyframes [Hong and Kim 2004; McNamara et al. 2004; Treuille et al. 2003] and moving target shapes [Fattal and Lischinski 2004; Shi and Yu 2005a,b]. In visual effects, many practitioners have used these or similar techniques to build fluid characters [Sachs et al. 2010; Trojansky 2008; Wiebe and Houston 2004]. Despite being useful for characters, these techniques often make naturalistic fluid animation look over-controlled and unrealistic.

Consequently, many authors aiming for naturalistic fluid focus instead on different techniques. Several approaches allow augmenting a low resolution simulation with high resolution detail [Kim et al. 2008; Thürey et al. 2006; Yuan et al. 2011]. Alternatively, [Mihalef et al. 2004] allowed artists to explore fast 2D fluid wave simulations to choose initial conditions for 3D fluid waves. In reverse, [Horvath and Geiger 2009] showed how procedural 3D particles could be used to great effect to drive film resolution camera-aligned 2D fluid simulations. While resolution enhancement and partitioning can be useful, they do not fundamentally solve the problem of improving the accuracy of a windowed simulation setup.

Toward this, several works consider handling the boundaries of simulation for control. [Rasmussen et al. 2004] used particles to enforce soft and hard controls in a grid-based liquid solver. They employ Neumann boundary conditions in a volumetric region around each kinematic particle. These are then used as sources, sinks or collision objects. While these are useful primitives for enforcing boundary conditions, they do not consider systematic unification of these controls.

We were inspired by [Nielsen and Bridson 2011], which suggests that low resolution simulation can guide higher resolution simulation. They provide an algorithm to codify the artist practice of hiding unnoticeable kinematic collision objects underwater for control. In particular, the low resolution simulation is eroded to a kinematic inner core that also measures how much fluid is entering and leaving the core. The surface of this inner core is the window between a non-simulated part of fluid and the simulated part of fluid, so it is actually more than a collision object. This has a huge computational advantage, because the number of simulated high resolution particles grows more like surface area rather than volume. While this paper is inspiring, its method for defining the inner core is relatively inflexible. It does not address the problem of interfacing with the wider ocean, and it doesn't consider the possibility of an artist controlling the window region explicitly.

Several techniques allow creating a 3D windowed simulation. Pixar's Brave used a low-resolution river simulation to seed a high resolution detail simulation around characters [O'Brien 2013]. [Thürey et al. 2006] two-way coupled 2D shallow water with a 3D solver. However, most practitioners cheat by simulating a boat in 3D on a flat plane of water and adding Tessendorf displacement post-sim. This works well if the boat is large compared to the waves or the waves are small compared to the boat. Reflecting waves are still a problem but can be handled by perfectly matched layers (PML) [Söderström et al. 2010]. While [Bojsen-Hansen and Wojtan 2016] generalized PML to non-flat procedural water, it only works for cubic domains. Recently, Autodesk's Bifröst framework (see e.g. [Nielsen and Bridson 2016]) provided a system for combining ocean simulations with near-field simulations, but details of the underlying algorithm are not published or described in detail.

3 MOTIVATION

Consider an art-direction problem: given an animated boat, a procedural ocean surface (Figure 3a), and a simulation window, produce a fluid simulation matching the ocean while also interacting with the boat (Figure 3b). Since we desire a naturalistic result, we choose a boundary condition approach rather than internal forcing. For simplicity, we first consider a windowed simulation without the boat (Figure 3c). That is, our simulation should match the procedural motion on the subdomain it replaces. For further simplicity, consider a calm and flat ocean (Figure 3d) with zero velocity and height. A popular and naïve approach defines the window as a box where the bottom and side walls are collision objects. When the simulation window moves, liquid particles are forced to move with the window rather than remain still. A correct approach instead sources material ahead of the window and sinks material behind the window (e.g. an Eulerian open water boundary). In Lagrangian simulation, velocities of the particles are initialized with a prescribed ocean velocity independent of the window motion.

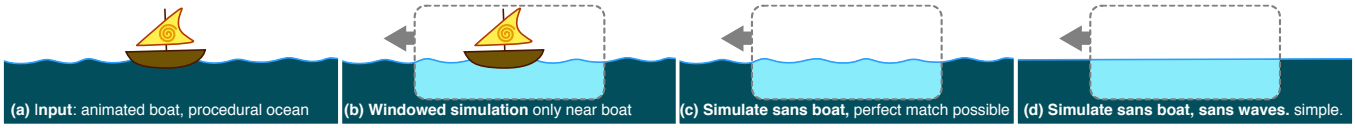

Fig. 3. Simulating a boat on an open water surface. ©Disney

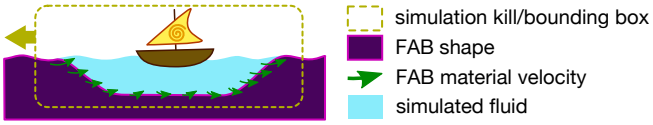

Fig. 4. An open water simulation setup with FAB method. ©Disney

Now, return to a more realistic procedural ocean (Figure 3c). If the procedural waves come from a valid [Tessendorf 1999] spectrum, they correspond to a physical motion of water, so our simulator should replicate the ocean, given correct velocity boundary conditions. Some parts of the boundary need to act as sinks, some as sources, some as kinematic solids. However, simply constraining the boundary to be the world velocity prescribed by the procedural ocean covers all cases. See Sections 4 & 5 for more details. In particular, we show how to obtain a full field of Tessendorf and Stokes velocities rather than merely surface velocities. Correct boundary velocities are key to preserving the ocean spectra structure.

Given this setup, we now put the boat back (Figure 3b). Its motion could be hand-animated, two-way coupled with the fluid, or produced via a buoyancy simulation. However defined, the motion will disturb consistency between the simulation and the larger ocean due to [Tessendorf 1999] assuming no obstacles. The effects of this mismatch are small but are further minimized by constructing the window shape to become shallow on the edge of the simulated region. The velocities near this shallow area may also be damped toward the velocity of the ocean. Extending PML [Söderström et al. 2010] to our irregular shapes would be interesting future work to investigate, but we found our simple damping worked sufficiently well for our needs.

4 OUR METHOD

Now, consider the practical details of our method. Creating a windowed simulation requires specifying the window's shape and the material velocity on its boundary for every point in time. However, part of the window may not be immersed in the material, as shown in Figure 3, so we limit the influence of the outside material to only certain parts of the window boundary. Specifically, an artist creates a set of control volumes called FABs that bound regions of the window containing material. They must have precise shape and material velocity at the boundary of the simulation window. Elsewhere, they can have arbitrary shape, e.g. truncated to have finite support. The simulated region then is the subset of space containing material (particles) but not contained in any FAB. Note that although material velocity is formally only required at the window boundary, in practice it is typically available throughout the whole volume of a FAB. This proves to be especially useful with finite timestepping, so we will assume a volumetric definition in what follows.

Figure 4 illustrates how a FAB can be used to set up a boat wake simulation from Figure 3b. As before, the fluid is simulated only in a bounded region around the boat, but now it is "contained" by a FAB. The shape of the FAB represents the outside of the simulated region around the boat. Note that only the material part below the ocean surface is represented, and that it only represents finite rather than infinite support. Finite support is obviously useful for efficiency and is not strictly required in general. Also a kill box may be employed to remove stray FLIP particles. We tend to make it somewhat wider than the simulation region. The FAB material velocity represents the motion of the ocean inside of the shape and is used to enforce the boundary conditions and for the material set update: sourcing and sinking.

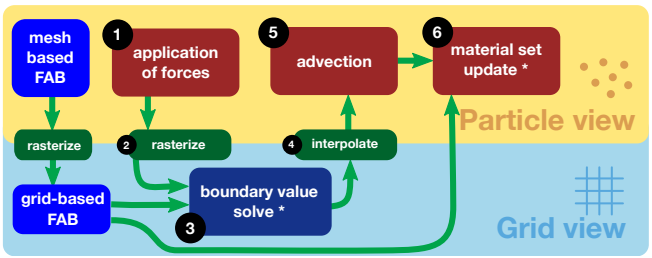

Fig. 5. FLIP/APIC/MPM solver time step structure augmented with FAB. Steps with the asterisk are the only ones affected by the FAB method. ©Disney

4.1 Integration within solver

We now describe how to integrate FABs into an existing FLIP, APIC, or MPM solver, for which typical high-level time integration steps are shown in Figure 5. A time step starts with applying forces, such as gravity, to particles. The particles are then rasterized to a grid, allowing a boundary value solve to update velocities. We use Poisson projection for inviscid fluids and a stress-based solve for solids and viscous fluids. Grid velocities are then interpolated back to particles; particles are then advected. Finally, the material set is updated by deleting unneeded particles and adding required new particles. Note our method is also easily adaptable to other simulation loops. In our loop, FABs only affect steps 3 and 6: *boundary value solve* and *material set update*.

During the **boundary value solve**, FABs provide a Neumann pressure boundary condition for Poisson projections and a Dirichlet velocity boundary condition for stress-based solves. This is analogous to kinematic solid boundaries [Bridson 2008], with the caveat that we sample the material velocity rather than the shape velocity.

The **material set update** adds and removes particles. Particles that end in a FAB after advection are deleted. Particles should be

seeded when material starts inside a FAB but is advected to be outside. [Nielsen and Bridson 2011] propose an approach to seeding for their low-resolution guide objects based on level sets. They advect the low-resolution guide object by the low-resolution fluid flow and seed in the level set difference region. We found this grid-based approach to introduce aliasing and accuracy problems, even with the dilation/erosion operations they suggest to compensate for these problems. We found a simple and accurate alternative. We create temporary particles inside the FABs at time t and advect them with the sampled material velocity. If after advection at time $t + \Delta t$ they end outside the FABs, we seed them as new active simulation particles, otherwise we discard them. This is illustrated in Figure 7.

4.2 Discussion

Given the FAB implementation, we can express many familiar simulation primitives using them. For example, if the normal component of the material velocity exceeds that of the velocity of the shape, the FAB would act as a source. If it is less, the result is a sink. If they match, the object acts as a kinematic solid. Thus, FABs can be considered as a generalization of well-known concepts of sources, sinks, and kinematic solids, alleviating the need for separate controls. An artist can work directly with the shape and material velocity field, and let the system handle the rest. Figure 6 shows typical behaviors FABs may exhibit during simulation.

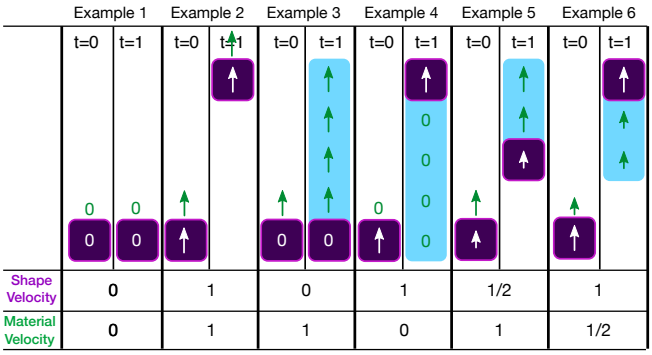

Fig. 6. Example behaviors exhibited by six non-deforming FABs with prescribed material/shape velocities. All FABs have the same position and shape at time $t = 0$ but different final time $t = 1$ configurations. Notice how the relative velocity between material and shape determines the sourcing behavior. The color encoding matches that of Figure 4. ©Disney

4.3 Efficient reseeding

While one can trivially seed particles into FAB shapes completely, it is much more efficient to take advantage of the CFL condition by only seeding in a thin band around the FAB surface. In practice, we use a VDB, see [Museth 2013], to represent the shape and perform seeding only through its active voxels. The VDB also gives us performant inside/outside tests.

The FAB shape and material velocity must be available at arbitrary times since solvers operate adaptively according to the CFL condition. Analytic kinematic shapes or ultra-fine pre-baking are not practical in production, where animation is done at coarse 24 FPS time steps. Thus, users provide a shape velocity (usually computed with finite differences), allowing advection of the shape VDB to an arbitrary time. Even so, advecting a high resolution VDB is expensive. Level set interpolation [Selle et al. 2008] can speed up inside tests for removing particles, but for seeding we need an efficient way to determine the sparse active set of voxels at the interpolated time. Thus, we propose a hybrid method. We downsample the level set and advect it efficiently. Upsampling this advected low-resolution result gives us a valid set of active voxels. Lastly we reinitialize its SDF values to be more accurate using level set interpolation.

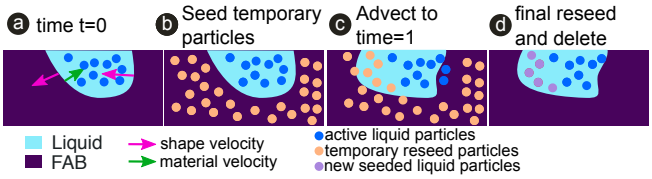

Fig. 7. Automatic FAB seeding and particle deletion. ©Disney

4.4 Creating a material velocity field

As discussed above, a material velocity field needs to be available for sampling in a thin band around the corresponding FAB, typically a few cells wide, depending on the CFL. When an analytic expression is unavailable, it is often practical to compute the material velocity on discrete particles. Accuracy is sufficient if the inter particle-spacing is commensurate with the solver grid cell size. If the material velocity computation is only available on the surface of the FAB, we simply use nearest neighbor sampling to extend it. Finally, particle velocities are rasterized to a VDB volume for efficient simulation-time lookup.

4.5 Artistic applications

Fig. 8. An example of artistic FAB application. ©Disney

The artists who tested FABs found the shape and material velocity controls to be very intuitive. They found many immediate uses, like creating the water creature shown in Figure 10. They started with a sculpted shape with no surface detail and painted curves on its surface that represented the desired material flow. A noise was applied to the curves to add interesting detail, and the tangents were sampled on tubes of particles around the curves and rasterized to a volume to create a material velocity field. The simulation was run with no gravity and had a very weak force field pointing towards the shape to avoid stray particles. The FAB method allowed the force field to be small, yielding a naturalistic rather than over-controlled look. Figure 8 shows a production image created with FAB.

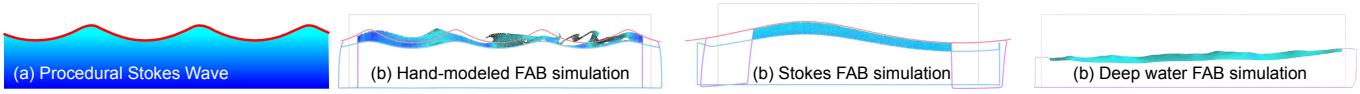

Fig. 9. Open ocean FAB simulations. Physical and accurate material velocity is required for matching the procedural surface. ©Disney

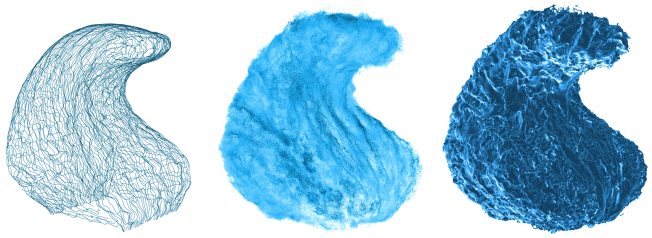

Fig. 10. Making a water creature: sculpted shape with material velocity curves painted by an artist (left), FLIP/APIC simulation with FAB (center) and a meshed result (right). ©Disney

5 PHYSICAL SIMULATION WITH FAB

The water creature example from Figure 10 demonstrated artistic use of FABs: the shape and the material velocity were driven creatively achieving a "magical" effect. Non-fantastical scenarios such as a boat on open water require the motion and hence the material velocity to be physically accurate. Otherwise, seaming and other artifacts will be present. We found that hand-modeling of material velocities was infeasible for an artist to perform with sufficient accuracy. In the following we present methods and techniques for producing two important types of FAB: Stokes waves and deep water waves.

5.1 Swells and Stokes waves

Swells are gravitational waves propagating along the ocean surface. They are typically generated by distant winds and may travel long distances without significant change in shape [Young 1999]. A Stokes wave is a good approximation for a swell [Stokes 1847]. It is a non-linear and periodic surface wave on a fluid layer of constant mean depth representing a potential flow solution to inviscid Navier-Stokes equation. An example of a Stokes wave is shown in Figure 9a. A closed-form representation for a Stokes wave is not known but several approximations exist. In particular, for an ocean of infinite depth [Dingemans 1997] provides

$$\Phi(x,z,t) = \frac{a\omega}{k} e^{kz} \sin(kx - \omega t), \tag{1}$$

with dispersion relation $\omega^2 = gk$ and the following notations

Φ	velocity potential, velocity $\boldsymbol{v} = \nabla\Phi$
x	horizontal coordinate
z	vertical coordinate, with the positive direction upward and $z = 0$ corresponding to the mean surface value
t	time
a	amplitude
k	wavenumber, $k = \frac{2\pi}{\lambda}$, where λ is the spacial wavelength
ω	frequency, $\omega = \frac{2\pi}{\tau}$, where τ is the temporal period
g	magnitude of gravity

Any combination of parameters a and λ leads to a unique and physically correct Stokes wave, provided it satisfies the steepness constraint

$$2\frac{a}{\lambda}\left(1 + \frac{3}{2}\left(\frac{a}{\lambda}\right)^2\pi^2\right) < .1412 \iff \frac{a}{\lambda} < 0.069, \tag{2}$$

which prevents excessive amplitude relative to wavelength. The constant on the right was determined numerically by [Michell 1893].

Unfortunately, the approximation is not a closed-form expression of the water surface—requiring us to derive more simplifications. The motion of a Lagrangian fluid particle in a Stokes wave is given by

$$\begin{cases} \dot{x} &= \frac{\partial\Phi}{\partial x} \equiv a\omega e^{kz}\cos(kx - \omega t), \\ \dot{z} &= \frac{\partial\Phi}{\partial z} \equiv a\omega e^{kz}\sin(kx - \omega t). \end{cases} \tag{3}$$

While integrating this numerically is possible, we desire a closed-form solution. Given (2), the wave amplitude is small compared to its length, allowing us to assume e^{kz} is nearly constant, yielding

$$\begin{cases} x(t) &= -ae^{kh}\sin(kx_0 - \omega t), \\ z(t) &= ae^{kh}\cos(kx_0 - \omega t), \end{cases} \tag{4}$$

where x_0 is the reference horizontal position and h is the mean depth of a Lagrangian particle, with h being negative in the fluid volume and 0 on the surface. The difference between (4) and numerical integration of the ODE (3) is insignificant, manifesting itself mainly in disregarding the Stokes drift [Stokes 1847]. In fact, we found nearly perfect matching results between procedural surface and FAB simulation in the absence of the boat. With the boat, the error is dominated by the presence of the boat and requires additional post-process blending. Above all, (4) being closed-form allows us to combine it with the deep water waves directly.

We now have all the ingredients to make a FAB. The material motion is given by (4). In practice, we seeded particles in a region, deformed them with (4), defined velocities using finite differencing and rasterized them to a volume. Alternatively, one could define the material velocity by analytically differentiating (4) with respect to t.

5.2 Deep water waves

While the most popular ocean wave model is the one given in [Tessendorf 1999], [Horvath 2015] recently introduced more variations on the wave spectrums. Even so, both papers merely advocate the use of different statistical distributions of wavelengths, while the basic idea remains the same: deep water oceanic waves are just a superposition of multiple Stokes wave approximations (4), or Gerstner waves if only the surface is considered, i.e. $h = 0$, see [Fournier and Reeves 1986]. A practical way to compute the deep water surface displacement is given in [Tessendorf 1999]. A Fourier transform produces multiple sine waves, and a post-sharpening turns those into Gerstner waves.

Previously, only the deep water surface was considered; however, to create a FAB, we need velocities (and hence displacements) at

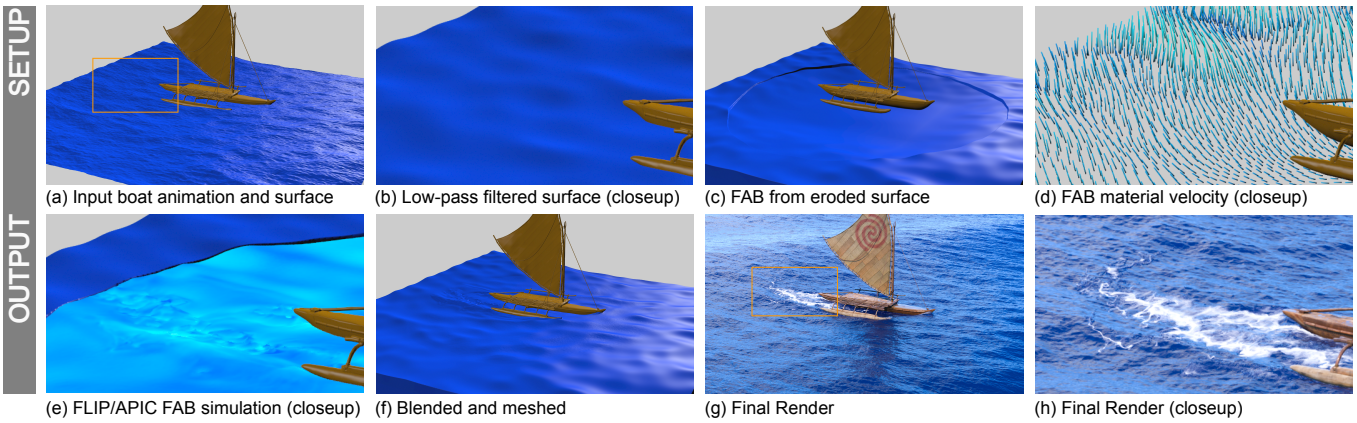

Fig. 11. Stages of simulating a boat on deep water waves. ©Disney

other heights. Computing displacement at mean depth h is accomplished by prescaling the frequency domain distribution coefficients by e^{kh}, similar to (4). However, Fourier transforms for many values of h would be too costly, so we found that linearly interpolating between only a few discrete depths was sufficiently accurate. Regardless of the computational approach taken, the final deep water expression is similar in spirit to (4), giving the displacement for Lagrangian particles of the fluid, and so material velocity computation can be carried out in exactly the same way as for the Stokes waves.

6 EXAMPLES

6.1 Simulating a swell

Now we will detail exactly how to create a windowed Stokes wave simulation using FABs. Using a deep water spectrum instead of a Stokes wave is completely analogous. We define the static window using a bounding box (Figure 12a), and we clamp it to the static water level, corresponding to $h = 0$ (Figure 12b). We then erode the box to get the FAB shape and make a cavity for simulated water (Figure 12c). To make a simulation on a Stokes wave, we simply apply transform (4) to the simulated region shape and the FAB shape (Figure 12d). Note, that the transform needs to be applied to the shape separately for all simulated frames, since it varies with time. Similarly, the material velocity field can be created by seeding auxiliary particles into the undeformed FAB (Figure 12e), deforming them with (4) to represent the actual FAB shape (Figure 12f) computing velocities using finite differences or by analytic differentiation, and rasterizing to a volume (Figure 12g). Figure 9b shows how inaccurate and non-physical FAB setups may cause a mismatch between a simulation and a procedural surface. This may occur for multiple reasons, such as inaccurate material velocity, violated steepness constraint (2), or wave frequencies exceeding the simulation grid's Nyquist limit. We found that the Stokes wave length should be at least several grid cells wide to avoid severe numeric dissipation. Figure 9c shows a properly set up FAB simulation with Stokes waves giving a perfect match with the procedural surface.

6.2 Boat wake on an ocean surface

As mentioned earlier, deep water waves such as the ones presented in [Tessendorf 1999] are just a superposition of multiple Stokes

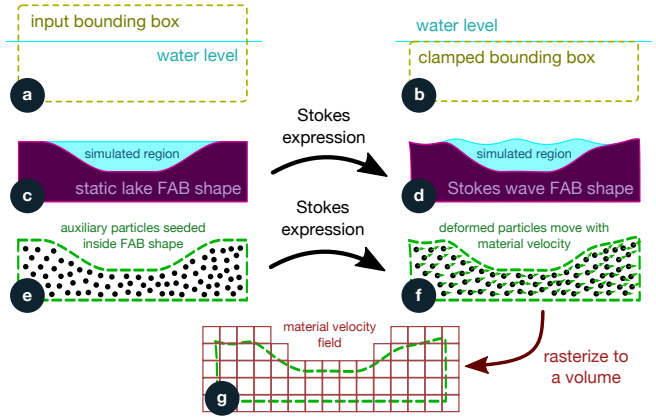

Fig. 12. A FAB simulation setup with Stokes waves: (a) input bounding box, (b) box clamped to height 0 (c) eroding the box gives FAB shape and simulated region for a static lake, (d) deforming with Stokes expression gives FAB shape and simulated region for a swell, (e) auxiliary particles seeded into the static lake FAB shape, (f) particles deformed with Stokes expression get material velocity, (g) material velocity rasterized from particles to a volume. ©Disney

waves, and hence a FAB simulation can be set up in a similar way. Figure 9d shows a FAB simulation for a superposition of several Stokes waves giving a perfect match with the procedural surface. It is worth emphasizing, that in order to have a physically correct deep water expression, each component of the spectrum must satisfy the steepness constraint (2). Otherwise, there are no guarantees of reproducing its behavior through simulation. Like in the Stokes wave example, a deep water spectrum may contain frequencies that exceed the Nyquist limit of the simulation FLIP/APIC grid, causing mismatch and aliasing. A practical way do deal with this is to low-pass filter such wavelengths before simulation and reapply as a post-process displacement later. Figure 11 shows all of the setup stages for simulating a boat on deep water waves.

6.3 Breaking wave

A breaking wave is fascinating. It starts as a swell away from shore, begins overturning as it gets closer, and finally becomes turbulent

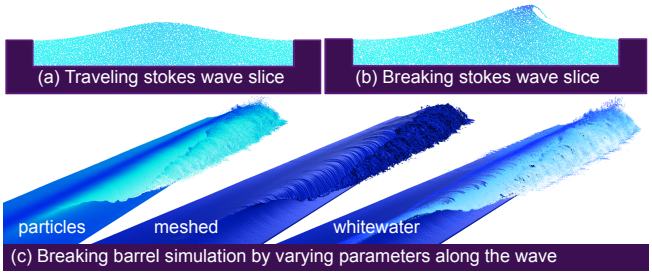

Fig. 13. Simulating a breaking wave with FABs. ©Disney

and chaotic. Typically, movie shots impose requirements on when and how a wave breaks, leading artists to use procedural rather than simulation tools to create them. Proceduralism, however, limits realism and complexity, forcing artists to mask lack of detail with artificial secondary passes. Thus, we propose a fully simulated approach to creating controllable breaking waves using FABs. We start by simulating a traveling swell. We simulate one isolated period of a Stokes wave. Since Stokes waves move with constant velocity, it is beneficial to perform the simulation in the frame of reference of the wave to avoid dealing with a moving window. Figure 13a shows a possible setup for the simulation: a period of a Stokes wave, surrounded by a static FAB. With this setup, the simulated wave would persist over time without changing its shape. To break the wave it suffices to change the boundary conditions to the ones that violate the steepness constraint, see Figure 13b. By varying the conditions along the wave we were able to achieve complex effects such as a surfing tube, breaking from one end, see Figure 13c. The render is shown in Figure 1 (top).

7 IMPLEMENTATION NOTES

7.1 Simulator modification

Modifying a simulator to implement our technique is easy to do, but we stress the key to its utility is making sure authoring material velocities is a first-class interface element.

User interface. We created a Houdini interface where artists authored deforming meshes that allowed finite-difference based shape velocities to be calculated. Artists could augment meshes with a material velocity on the mesh or by using free-particles. In either case, the material velocity would then be rasterized to a VDB.

Pressure projection modification. We modified the velocity projection to sample material velocities instead of shape velocities at points within the kinematic solid.

Seeding modification. We added a new seeding scheme that replaced the ad hoc particle creation [Reeves 1983] techniques used before.

FAB examples. We provided Stokes and Deep Water examples to artists. Lead artists then created rigs that eased repeated application of similar situations. In addition, sources and sinks were implemented in terms of FABs within the interface.

7.2 Blending and meshing

Blending and meshing are important to achieve final seamless integration with the far field. One of the goals of our method is to enable getting good results with the simplest blending techniques available. One key strategy for improving blending is preconditioning final simulated particles to match the far surface. This ensures that when the particles are meshed, outlier particles do not perturb the surface. Particles were meshed using [Yu and Turk 2010] and unioned with the FAB volume (Figure 11f). At render-time, we created a frustum aligned voxel grid and resampled the FAB and the far-field height field to a grid. At this point there was a tiny mismatch due to the heightfield only approximating the signed distance, so we alpha blended near the FAB boundary to get final signed distances yielding the final render surface (Figure 11g).

7.3 Deep water spectrum

Deep water methods [Horvath 2015; Tessendorf 1999] typically specify coefficients of the water spectrum in the frequency domain and use a Fourier transform to obtain displacement. This sum of sine-waves is then sharpened to obtain Stokes waves. Band-pass filtering operations in the frequency domain are trivial, requiring only setting the undesired bands to 0. For the boat wake setup, we filtered frequencies higher than the simulation grid resolution Nyquist limit because those frequencies would be damped out by the simulation (Figure 11b). The spectrum containing only the removed frequencies was reapplied to the meshed result at render time, Figure 11h.

A well-known problem with spectral deep-water methods are *tiling artifacts*. Computational and memory limitations forced our tiles to be only 2^{10} grid cells, yielding a 2^{10} feature size ratio e.g. (1km:1m). If the tile is too big (1km), the smallest wave features are too coarse (1m), if the tile is too small (10m:1cm), you see visually apparent copies of the same tile if you look at a (1km) region. Typically practitioners add different sized hand-chosen tiles, but avoiding artifacts on all scale shots is tedious and error-prone. Instead, we implemented an automated tile selection scheme. An artist chooses the largest area that will be visible in the camera (say 1 km) and the smallest needed feature size (say 1cm). Then, several 2^{10} sized tiles are created to cover the limits (Figure 14). To avoid double-counting in the overlap regions high and low pass filtering was performed accordingly, and none of the frequencies outside of the big tile were considered in the computation. This method proved to be significantly cheaper at the expense of not covering certain frequencies at all but in practice gave good visual results, see Figure 1.

8 CONCLUSION AND LIMITATIONS

In conclusion, we created the FAB method, unifying several disparate controls. We showed how to build artistic FABs to create characters and physical FABs to match procedural water. Before FABs, artists in our organization were frustrated by limited and ad hoc particle-based techniques, being forced to layer fixes on off-the-shelf control techniques. FABs were by far their favorite feature in our new fluid solver, allowing them to create hundreds of water shots in Disney's Moana (see our video and Figure 1).

In the future, we would like to use FABs more widely. While our method trivially works on elastoplastic MPM solves [Stomakhin et al. 2013], it would be interesting to create a library of physical elastoplastic procedural models. One limitation of our method is that creating physical velocities for FABs requires care, but we stress

Fig. 14. Multi-tile deep water spectrum construction. ©Disney

the situation is immensely better since we create a framework where physicality is more easily achieved. Additionally, while our method matches deep water and Stokes waves, another limitation is that a perfect match is not guaranteed once internal boundaries (like boats) are introduced. One idea would be to adapt perfectly matched layers to work on non-rectangular domains so that they could be used in concert with FABs. Regardless, we expect our technique would be a useful addition to any fluid solver.

ACKNOWLEDGMENTS

We appreciate our artistic users, specifically David Rand and David Hutchins for artistic and physical FAB examples, respectively. Ralf Habel and Patrick Kelly helped considerably with level set compositing and render-time meshing. We would also like to thank Lawrence Chai and Rajesh Sharma for initial discussions about improving the water simulation pipeline. Rick Hankins at ILM also provided a lot of initial insight and inspiration. Lastly, we would like to thank the Walt Disney Animation Studios leadership.

REFERENCES

Morten Bojsen-Hansen and Chris Wojtan. 2016. Generalized non-reflecting boundaries for fluid re-simulation. *ACM Trans. on Graph.* 35, 4 (2016), 96.

R. Bridson. 2008. *Fluid Simulation for Computer Graphics.* Taylor & Francis. https://books.google.com/books?id=gFI8y87VCZ8C

M.W. Dingemans. 1997. *Water Wave Propagation Over Uneven Bottoms: Non-linear wave propagation.* Number pt. 2 in Advanced series on ocean engineering. World Scientific Pub. https://books.google.com/books?id=FLTfoAut5dwC

Raanan Fattal and Dani Lischinski. 2004. Target-driven Smoke Animation. *ACM Trans. Graph.* 23, 3 (Aug. 2004), 441–448. DOI: https://doi.org/10.1145/1015706.1015743

Nick Foster and Dimitris Metaxas. 1997. Controlling Fluid Animation. In *Proc. 1997 Conf. on Comp. Graphics Intl. (CGI '97).* http://dl.acm.org/citation.cfm?id=792756.792862

Alain Fournier and William T. Reeves. 1986. A Simple Model of Ocean Waves. *SIGGRAPH Comput. Graph.* 20, 4 (Aug. 1986), 75–84. DOI: https://doi.org/10.1145/15886.15894

Jeong-mo Hong and Chang-hun Kim. 2004. Controlling fluid animation with geometric potential. *Comp. Anim. and Virtual Worlds* 15, 3-4 (2004), 147–157. DOI: https://doi.org/10.1002/cav.17

Christopher Horvath and Willi Geiger. 2009. Directable, High-resolution Simulation of Fire on the GPU. *ACM Trans. Graph.* 28, 3, Article 41 (July 2009), 8 pages. DOI: https://doi.org/10.1145/1531326.1531347

Christopher J. Horvath. 2015. Empirical Directional Wave Spectra for Computer Graphics. In *Proc. 2015 Symp. on Digital Production (DigiPro '15).* 29–39. DOI: https://doi.org/10.1145/2791261.2791267

Chenfanfu Jiang, Craig Schroeder, Andrew Selle, Joseph Teran, and Alexey Stomakhin. 2015. The Affine Particle-in-cell Method. *ACM Trans. Graph.* 34, 4, Article 51 (July 2015), 10 pages. DOI: https://doi.org/10.1145/2766996

Theodore Kim, Nils Thürey, Doug James, and Markus Gross. 2008. Wavelet turbulence for fluid simulation. In *ACM Trans. on Graph.*, Vol. 27. ACM, 50.

Antoine McNamara, Adrien Treuille, Zoran PopoviÄĞ, and Jos Stam. 2004. Fluid control using the adjoint method. *ACM SIGGRAPH 2004 Papers, SIGGRAPH 2004* (2004), 449–456. DOI: https://doi.org/10.1145/1186562.1015744

J.H. Michell. 1893. *The Highest Waves in Water.* https://books.google.com/books?id=MCgzAQAAMAAJ

Viorel Mihalef, Dimitris Metaxas, and Mark Sussman. 2004. Animation and Control of Breaking Waves. In *Proc. 2004 ACM SIGGRAPH/EG Symp. on Comp. Anim. (SCA '04).* 315–324. DOI: https://doi.org/10.1145/1028523.1028565

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

Michael B. Nielsen and Robert Bridson. 2011. Guide Shapes for High Resolution Naturalistic Liquid Simulation. *ACM Trans. Graph.* 30, 4, Article 83 (July 2011), 8 pages. DOI: https://doi.org/10.1145/2010324.1964978

Michael B. Nielsen and Robert Bridson. 2016. Spatially Adaptive FLIP Fluid Simulations in Bifrost. In *ACM SIGGRAPH 2016 Talks (SIGGRAPH '16).* ACM, New York, NY, USA, Article 41, 2 pages. DOI: https://doi.org/10.1145/2897839.2927399

Michael O'Brien. 2013. Running Rivers. *XRDS* 19, 4 (June 2013), 30–33. DOI: https://doi.org/10.1145/2460436.2460447

N. Rasmussen, D. Enright, D. Nguyen, S. Marino, N. Sumner, W. Geiger, S. Hoon, and R. Fedkiw. 2004. Directable Photorealistic Liquids. In *Proc. 2004 ACMSIGGRAPH/EG Symp. on Comp. Anim.* 193–202. DOI: https://doi.org/10.1145/1028523.1028549

William T Reeves. 1983. Particle systems–a technique for modeling a class of fuzzy objects. *ACM Trans. on Graphics* 2, 2 (1983), 91–108.

I. Sachs, C. Twigg, L. Uren, D. Pearson, and N. Rasmussen. 2010. Waterbending: Water effects on "The Last Airbender". In *ACM SIGGRAPH 2010 Talks.*

Andrew Selle, Michael Lentine, and Ronald Fedkiw. 2008. A mass spring model for hair simulation. *ACM Trans. on Graph.* 27, 3 (2008), 64.

Lin Shi and Yizhou Yu. 2005a. Controllable Smoke Animation with Guiding Objects. *ACM Trans. Graph.* 24, 1 (Jan. 2005), 140–164. DOI: https://doi.org/10.1145/1037957.1037965

Lin Shi and Yizhou Yu. 2005b. Taming Liquids for Rapidly Changing Targets. In *Proc. 2005 ACMSIGGRAPH/Eurographics Symp. on Comp. Anim.* ACM, 229–236. DOI: https://doi.org/10.1145/1073368.1073401

Andreas Söderström, Matts Karlsson, and Ken Museth. 2010. A PML-based nonreflective boundary for free surface fluid animation. *ACM Trans. on Graph.* 29, 5 (2010), 136.

G.G. Stokes. 1847. On the theory of oscillatory waves. *Trans. of the Cambridge Philosophical Society* (1847).

Alexey Stomakhin, Craig Schroeder, Lawrence Chai, Joseph Teran, and Andrew Selle. 2013. A Material Point Method for Snow Simulation. *ACM Trans. Graph.* 32, 4, Article 102 (July 2013), 10 pages. DOI: https://doi.org/10.1145/2461912.2461948

Jerry Tessendorf. 1999. Simulating ocean water.

N. Thürey, R. Keiser, M. Pauly, and U. Rüde. 2006. Detail-preserving Fluid Control. In *Proc. of the 2006 ACM SIGGRAPH/Eurographics Symp. on Comp. Anim.* 7–12. http://dl.acm.org/citation.cfm?id=1218064.1218066

Nils Thürey, Ulrich Rüde, and Marc Stamminger. 2006. Animation of open water phenomena with coupled shallow water and free surface simulations. In *Proc. 2006 ACM SIGGRAPH/EG symp. on Comp. Anim.* Eurographics Association, 157–164.

Adrien Treuille, Antoine McNamara, Zoran PopoviÄĞ, and Jos Stam. 2003. Keyframe control of smoke simulations. *ACM Trans. on Graphics* 22 (7 2003), 716–723. DOI: https://doi.org/10.1145/882262.882337

Stephan Trojansky. 2008. Raging Waters: The Rivergod of Narnia. In *ACM SIGGRAPH 2008 Talks (SIGGRAPH '08).* Article 74, 1 pages. DOI: https://doi.org/10.1145/1401032.1401127

Mark Wiebe and Ben Houston. 2004. The Tar Monster: Creating a Character with Fluid Simulation. In *ACM SIGGRAPH 2004 Sketches (SIGGRAPH '04).* 64–. DOI: https://doi.org/10.1145/1186223.1186303

I.R. Young. 1999. *Wind Generated Ocean Waves.* Elsevier Science. https://books.google.com/books?id=ph7GKZZGjyYC

Jihun Yu and Greg Turk. 2010. Reconstructing surfaces of particle-based fluids using anisotropic kernels. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '10).* Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 217–225. http://dl.acm.org/citation.cfm?id=1921427.1921459

Zhi Yuan, Fan Chen, and Ye Zhao. 2011. Pattern-guided Smoke Animation with Lagrangian Coherent Structure. *ACM Trans. Graph.* 30, 6, Article 136 (Dec. 2011), 8 pages. DOI: https://doi.org/10.1145/2070781.2024170

Received January 2017; accepted April 2017