STAG: Smart Tools and Applications in Graphics (2025)
M. Comino Trinidad, C. Mancinelli, D. Cabiddu, and D. Giorgi (Editors)

Implicit Field-Based Stylization of 2D and 3D Liquid Animations

Rodrigo Stevenson-Regla[1], Damien Rohmer[1], Loïc Barthe[2] and Marie-Paule Cani[1]

[1]LIX, École Polytechnique, CNRS, IP Paris
[2]IRIT, Université Toulouse III, CNRS

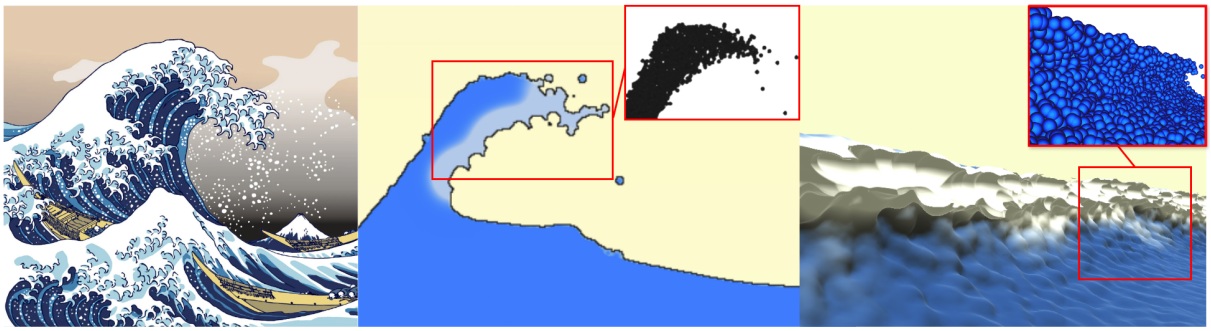


Figure 1: *Comparison between the intended expressive representation of breaking waves (left), and the 2D and 3D results obtained by stylizing input simulations (the rendered 2D or 3D simulation step is shown as inset image).*

Abstract

Combining physically-based simulation with stylized visual effects not only requires to change the aspect of surfaces, but their shapes as well. We describe a new expressive rendering method for liquid animations, which can be used on top of any preexisting particle-based simulation. Our solution builds on visual particles that carry both water and air distributions, both evolving through particle history based on kinematics information from the simulation. These density fields are combined at each frame to create the implicit iso-surface of interest, rendered in adapted style. By defining series of visual particle states, we parametrize this model to capture the typical stylized geometry of water bodies used to highlight dynamic motion in paintings and cartoons, such as elongating droplets, concavities carved at the crest of breaking waves, and stylized air-water mixtures such as bubbles and foam. Regardless of the 2D or 3D nature of the input simulation, our solution maintains temporal coherence and ensures that water bodies keep an approximately constant surface in 2D, resp. or volume in 3D, over time. Finally, we conducted a user study to show the effectiveness of our method against state of the art AI-based tools, in a variety of animation scenarios where stylized shapes are needed.

1. Introduction

Water animations have been used for many years in movies, cartoons, video games, and scientific visualizations. While most research works focused on the modeling of the realistic behavior of liquids, stylized cartoons and paintings rather depict liquids using exaggerated representations featuring typically elongated droplet shapes, or bulks associated with foamy surfaces (see Figure 2). These details correspond to changes of geometry of the liquid surface, aimed at either better conveying dynamic motion - such as for droplets and breaking waves - or the local nature of the fluid (such as for bubbles or foam, where water is mixed with air). Rather than modeling a realistic behavior, the aim of these effects is to provide an expressive representation, helping the viewer to intuitively feel the dynamics of the motion, or the nature of the fluid. Such expressive representations remain largely overlooked by both 2D and 3D fluid animation models in the research literature.

In this work, we explore the possibility of representing such stylized effects from standard physically-based simulations of liquids. Therefore, starting from such a simulation result, we define a set of *visual particles*, guided by their simulation counterparts, and able to carry two local density fields respectively representing the presence of liquid and of air around their position. Their distributions vary over time based on the particle's history and kinematic properties, while maintaining the amount of carried liquid to a constant value. The surface to display is obtained as an isosurface of the global field formed by summing all individual particle contribu-

DOI: 10.2312/stag.20251324

Figure 2: *Examples of artistic representations of water in both artistic paintings and cartoon animations. We identified four main effects: (a) Expressive droplets, (b) Concave shapes, (c) Bubbles and (d) Foam. We provide both a large scene and a close-up image for each of them.*

tions, but where the field around each of them is already a combination of the different stylization effects. By adjusting the shape of the parametric fields, the user can easily increase or decrease the emphasis placed on each effect, while always maintaining temporal coherence through visually continuous shape changes, including possible changes in topological genus.

Our contributions are threefold: First, we propose the notion of hybrid water-air visual particles, guided by the simulation but controlled by a state machine that triggers the appearance and transition between stylization effects. Second, we detail how this representation can be used to cover four different effects identified in preexisting stylized liquids in art and cartoons, namely droplets, concave effects, bubbles and foam. Third, we introduce a versatile way to combine the effects in terms of final shape and appearance of the water surface, while providing intuitive control to the user.

These contributions are integrated in a post-processing tool that only requires simple information from a simulation to produce the desired result. Focusing on expressive shape modeling and non-photo-realistic rendering, our tool acts as a bridge between particle computations and stylized visualizations. An illustration, showing both the 2D and 3D versions of the tool, is provided in Figure 1.

2. Related Work

Water-air interactions are essential to achieve detailed and visually appealing representations of water, with foam and bubbles, as exemplified in recent video games and movies [DHV*23, ACCK18]. Although some earlier models have simulated two-phase fluids to explicitly model air-water interaction, they cannot be used affordably on a large scale—such as to generate the bubbles and foam created by an ocean wave. Therefore, liquid simulation methods in Computer Graphics focused on water-only simulation, requiring to model air entrapment as a second step (see [Bri15] for a full review). Ihmsen et al. [IAAT12] defined a criteria with three characteristics to measure air entrapment. Bender et al. [BKKW19] improved this by adding vorticity as a fourth characteristic, Wretborn et al. [WFS22] proposed an aeration criteria based on particle stress. Baek et al. [BUH15] showed interaction between solids and liquid for muddy water. Ihmsen et al. [IBAT11] elaborated a method to animate air bubbles with SPH methods. Wretborn et al. [WSB25] proposed a method for diffuse bubbles for different models of bubbles distributions.

While the methods above aimed at photorealistic rendering of simulated liquids, artworks and cartoons introduced a variety of stylized water representations, which often heavily differ from the real phenomena, see Figure 2 . For example, while water droplets in real life are roughly spherical, in animation, it is common to represent them as a combination of a half-sphere with a trailing tail, creating a complex shape. Expressive representations of liquids therefore attracted some attention. Eden et al. [EBG*07] presented a method for rendering water based on cartoon animations. Liao et al. [LYP11] depicted ocean waves based on drawn animation styles. Liu and Zhang [LZ13] used elliptical shapes to model splashes when interacting with ships or rocks. Neto and Apolinario [RNA14] proposed a model for bubbles and foam for air-water interactions in video games, adopting a non-realistic style. The same year, Browning et al. [BBRF14] introduced a method that uses key frames generation to portray hand-drawn fluids. Lee [LKLK19] uses anisotropic shapes to represent droplets, though this method results in minor deformations which differ from the more exaggerated artistic styles. In comparison, our work aims to provide a unified method to automatically generate a set of stylized effects like the elongated droplets or the concave shapes.

Recently, a neural solution was proposed by Kim et al. [KAGS20], where painting-like textures were advected over liquid surfaces using Lagrangian neural networks. In contrast, our goal is to generate an explicit change of shape for the simulated liquid, be it 2D or 3D. More recent neural solutions [KAOT23, TKAS23, DYZ*23] achieved both changes of visual style and of geometrical shape, but mostly focused on fire and smoke. Their

method is therefore not applicable to water, or water and air mixtures. In summary, as shown by a recent survey [CSS*25], stylization of liquids still remains a challenging task both in terms of availability of the training databases and user control on the desired effects.

In this work, we build on the versatility of implicit modeling [BBB*97], a perfect representation for animating highly deformable shapes that split and merge over time [CD97, OC97, CGB16].

Another source of inspiration is sketch-based animation [CSGC16, ATW*17], where advanced blending operators such as those proposed by Gourmel et al. [GBC*13] play a key role in generating complex shapes with easily controllable features. Several solutions were designed for the stylized visualization of fluids: Kazi et al. [KCG*14] generate animations by adding coordinated motions to collection of objects in illustrations. Similarly, Jamriška et al. [JFA*15] introduced a method to create sprites from static images through flow-guided texture synthesis. Xing et al. [XKG*16] set up an interface to create stylized effects based on user controlled velocity fields. Ma et al. [MWK22] developed a system to create stylized 3D effects focusing on ease of use and expressiveness. A sketch-based animation system dedicated to stylized 2.5D biological environments enabled to animate cells carried by a fluid [OBG*24]. Lastly, Xie et al. [XASM24] proposed a sketch-based interface building on Lagrangian style transfer to create smoke illustrations.

3. Visual Particles and their State Changes

3.1. Notion of Visual Particles

Our method relies on visual particles, coupled in terms of position to the water particles of the simulation, but designed to carry a series of characteristics dedicated to the expressive deformation and rendering of the scene. Throughout the simulation, any change of parameter of the simulated particles is immediately reflected into a similar change for their visual counterpart. In reverse the visual particles never affect the simulated ones. This enables us to use our expressive visualization method on top of any pre-computed particle-based liquid animation. In practice, to avoid recomputing many parameters from the positions only of simulated particles, we extract a few relevant particle parameters at each frame of the simulation (position, velocity, density, neighborhood list and normal), and reuse them for the associated visual particles when we reload a given frame at the rendering stage.

To achieve the visualization we are aiming for, the visual particles are modeled as hybrid volumes carrying local distributions of both water and air. This allows us to mimic the stylized effects of air-water interaction without adding any extra simulated particle. To this aim, the virtual particles are driven by an effects state machine, detailed in Section 3.3, designed to produce the different stylization effects we identified. These include elongated water droplets, concave shapes at the edge of breaking waves, as well as stylized air bubbles and foam.

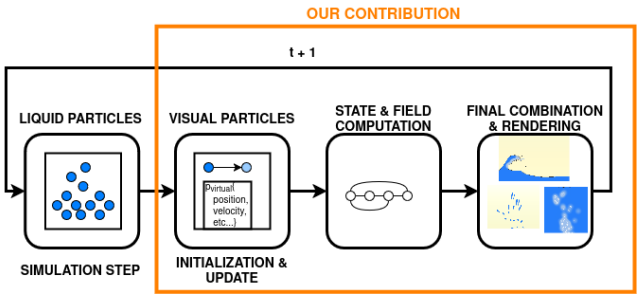


Figure 3: *Processing pipeline of our method, where our contribution is highlighted. After each simulation step (left) we either initialize (first step) or update each visual particle, compute their effect-state and the associated air and water density fields (middle), and finally combine their contributions to render the current frame in an expressive way (right).*

3.2. Processing Pipeline

Our processing pipeline is presented in Figure 3: At each time step of the input liquid simulation to be rendered, we first create or update the positions and other parameters (velocity, normal, density, etc) of the visual particles based on their simulated counterparts. Second, we employ a state machine to identify and control the local visual effect for each specific visual particle. Enforcing sequential state transitions within the state machine guarantees the visual temporal coherence as explained in Section 3.3. Based on the visual effect indicated by the state, we deduce properties such as material density distributions for water and air, modeled as two scalar fields around the particle's center. Third, we apply a specific rendering process to the implicit shape defined by combining the visual particle's scalar field in order to generate the final, stylized image.

Note that this pipeline can be used regardless of the dimension of the simulation: It can be applied, with only minimal change on the effects, either to render 2D simulations into flat cartoon-like style (see Figure 1, middle), as well as to produce stylized renderings of 3D animations (Figure 1, right). The generation of the 2D, versus 3D version of each visual effect is discussed in Section 4.

3.3. Changes of State

The visual effects produced by the virtual particles are controlled by a state machine. In this work, although more effects could be added in the future, we focus on the four visual effects identified in Figure 2. Our state machine, depicted in Figure 4, therefore has four states: the droplet state $S_{droplet}$ used for water-only particles, the concave state $S_{concave}$ targeting stylized concave regions at the crest of breaking waves, the bubble state S_{bubble} used for undersea bubbles, and finally the stylized foam state S_{foam}. These states produce the different effects for the final visualization.

From droplet to concave state. All particles start with their state machine initialized at state $S_{droplet}$, used to represent both water bodies and expressive droplets. A particle may then immediately switch to state $S_{concave}$ if it is detected to be at the crest of a wave.

Particles are considered to belong to a wave crest when the liquid

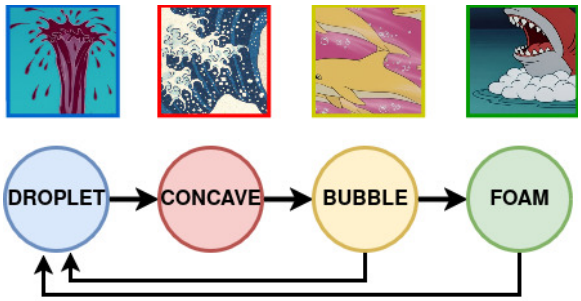


Figure 4: *The Effects State Machine. A particle can only transition from a state to one of its successors in this oriented graph. Each state encodes a given visual effect, and each oriented edge comes with a set of conditions for applying the transition.*

surface they contribute to exhibits high surface curvature. We use the wave crest detection method defined in [IAAT12] associated with the estimation of a local curvature measure κ_i for a particle i. A particle is considered to belong to a wave crest when $\kappa_i > \kappa^*$, where κ^* is the curvature threshold. When this condition is met, the particle transitions from state $S_{droplet}$ to $S_{concave}$.

Once in state $S_{concave}$, a surface visual particle starts to absorb air. We model this absorption from the volume of air the particle intersects during a time-step, given its velocity, inspired by the criteria defined in [IAAT12]. In our implementation, however, we set a maximum amount of air volume $V_{i,air}$ a particle can absorb, to match the targeted rendering style (see Section 4).

From concave to bubble state. A particle is considered to remain in a concave state until a significant increase in local water density is observed, indicating that it has entered a denser region of the fluid. While there is a variety of methods to detect such change, our implementation follows the approach of Ihmsen et al. [IAAT12], which defines a dense fluid region as one where particles and their neighbors exhibit similar velocities, indicating a homogeneous flow. In practice, we consider a change of state from $S_{concave}$ to S_{bubble} when the difference of velocities between the particle and its neighbors is below a given threshold.

Once in state S_{bubble}, the advection of air within the fluid is modeled by propagating the air quantity from one particle to its neighbors along the air flow direction. The initial particle then transitions back to $S_{droplet}$ (as it no longer contains air), while the neighboring particle switches to S_{bubble}. This mechanism enables us to render rising bubbles, whose precise trajectory is computed by finding the best candidate to propagate the air given the neighborhood of the initial bubble particle and a water flow direction. In our implementation, we were inspired by [BKKW19] to model such underwater air flow, while also providing two parameters under the user's control. This is formulated as follows:

$$\begin{aligned} f_d &= \Delta t \left(-k_b g + \frac{k_d}{\Delta t}(\vec{v}_i - \vec{v}_i) \right) \\ p_c &= \min_{j \in P, j \neq i} (d(x_i + f_d, x_j)), \end{aligned} \tag{1}$$

where f_d is the air flow direction, k_b is the buoyancy coefficient, g is the gravity, k_d is the drag coefficient and $\vec{v}_i$ is the local velocity of particle i. In this process, the air particle is transferred to the particle p_c closest to a target position defined by the particle position x_i and the air flow direction f_d.

From bubble to foam. Once a visual particle in state S_{bubble} reaches the surface it switches to state S_{foam}. This is detected from the advection itself, namely if the next candidate in the advection process is not in the direction of the flow, then the surface is considered to be reached. A particle at state S_{foam} does not transfer the air it contains anymore but it can still receive air from other particles at state S_{bubble}. Particles at state S_{foam} have a minimum lifetime and when it is reached, they can burst, releasing all their air and going back to $S_{droplet}$. This process is inspired by the lifetime defined in [WFS22] where the lifetime of the foam is related to the amount of neighboring foam.

4. Stylized Effects

While their changes of state were inspired by former work, visual particles now need to be used in novel ways to achieve the expressive rendering effects we aim to achieve. Among the possible artistic representations, we decided to focus on the four typical effects illustrated in Figures 2, chosen for their expressivity in animation and their aesthetic. Our solution builds on implicit surface representations. In this section, we explain how visual particles are used, based on their state, as skeletons generating water and air density fields around them. These fields are finally combined, as detailed in Section 5 to generate implicit surfaces representing the stylized shapes we are seeking.

4.1. Expressive Droplets

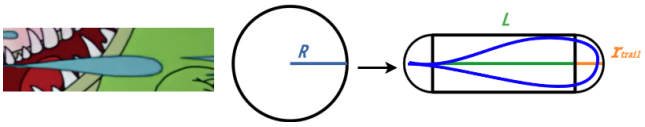


Figure 5: *Based on cartoon representations of water (left), expressive droplets are composed of a circular (resp. spherical in 3D) head and a tail of decreasing radius, which center-line expresses the droplet's trajectory in the last time steps.*

Our first goal is to create a representation for expressive droplets, inspired by the examples in Figure 2 (a) and 5 (left). In these pictures, droplets are depicted as elongated shapes that resemble the combination of a circle with a curved or straight tail that captures the recent droplet's trajectory. The length of the tail seems to depend on the droplet's speed, as shown by some nearly circular droplets on top of Figure 2 (a).

To achieve this effect both in 2D and 3D (depending on the nature of the input simulation), we define the implicit surface representation for any visual particle in $S_{droplet}$ state by creating a water density field - a scalar function decreasing with the distance - around a skeletal poly line generated from the trajectory of the particle in the few last time-steps (see Figure 5, right). The skeleton

joints J_i are computed as follows:

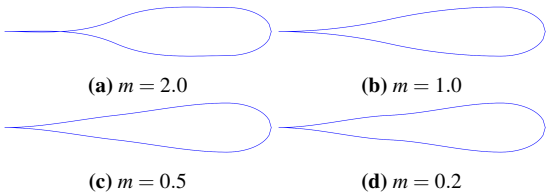
$$\begin{aligned} j_i^0 &= x_1^0, x_i^t \in X_i^T \\ x_i^t \in J \rightarrow d(j_i^{k-1}, x_i^t) &\geq \epsilon^*_{\text{joints}}, t \in [1,T] \end{aligned} \tag{2}$$

where X_i^T is the set of positions of a particle i at the previous T frames, and d is a distance function. We use the particle's current position as the initial joint. While the choice of the constant T (number of previous positions used) already controls the length of the trajectory based on the particle speed, clamping the skeleton to a maximum length may be needed to achieve the desired style. Therefore, we use ϵ^*_{joints} as a distance threshold along it.

We now need to choose an adequate distribution of water along this skeleton, to generate the desired shapes of decreasing radius (see Figure 5 (right)). This is done by specifying a water-density field function, designed such that the implicit surface of interest – 0.5 iso-surface in our implementation – takes the adequate shape. A key constraint is that, regardless of the tail's length, the droplet should maintain a constant surface area in 2D or volume in 3D, as the total amount of water remains unchanged over time.

To achieve this, we express an approximated preservation of area (resp. volume) between the droplet at rest, i.e. a circle (resp. a sphere), and a simplified encapsulating shape of the droplet made in assembling a circle (resp. sphere) for the head, and a rectangle (resp. cylinder) for the tail. This relation is expressed in Eq. (3), where r_{trail} is the radius attached to the head join, R is the radius of the circular (resp. spherical) droplet at rest, and L is the length of the skeleton. The solution is obtained by solving (3) under the constraint that $0 < r_{trail} \leq R$.

$$\begin{aligned} \pi R^2 &= \pi r_{\text{trail}}^2 + 2Lr_{\text{trail}} \text{ in 2D} \\ \frac{4}{3}\pi R^3 &= \frac{4}{3}\pi r_{\text{trail}}^3 + \pi L r_{\text{trail}}^2 \text{ in 3D} \end{aligned} \tag{3}$$



(a) $m = 2.0$ **(b)** $m = 1.0$ **(c)** $m = 0.5$ **(d)** $m = 0.2$

Figure 6: *Different styles of trails obtained by changing the middle point tangent m of the spline used for skeletal radius variation.*

Once r_{trail} is defined, we compute the curved tail as a cubic spline defining a shape with decreasing radius r_s along the droplet's skeleton. This spline is defined to provide a smooth transition, with flat tangents at the extremities, between the largest radius r_{trail} to be used at the tip of the skeleton and a zero value at the far end. The varying radius r_s is then used to scale distance computations in the field function ϕ, a decreasing function of the distance, used to define the implicit surface of the droplet:

$$\phi(h) = \left(2h^3 - 3h^2 - 1\right)\phi_{\max} \tag{4}$$

where $h = d/r_s$, and d is the distance from a point in space to the closest point along the skeleton, and ϕ_{max} controls the maximum field value. We set $\phi = 0$ for $h > 1$ to avoid any undesired bulk away from the visual particle. The implicit surface is obtained as the iso-value 0.5 of ϕ.

Figure 6 shows how the shape of the implicit surface can be tuned by adapting the spline curve that defines the radius: While tangents at the extremes are always set to 0, the use of a cubic spline enables us to control the tangent at the middle point along the skeleton, and obtain the different depicted shapes. We found that a middle tangent within $[0.2, 1.2]$ produces tails that resemble the most those of Figure 5.

Note that this method is easily applicable in both 2D and 3D cases, the only difference being the area or volume constraint on r_{trail} in Equation (3). A few results are shown in Figures 5 and 18, and 11 top right. As faster particles travel a larger distance between their previous positions, they generate longer skeletons resulting in a long, possibly curved tail, while particles that do not move much (such as those at rest within a large water bodies) resemble a circle in 2D or a sphere in 3D, and keep the radius R.

4.2. Concave Effects

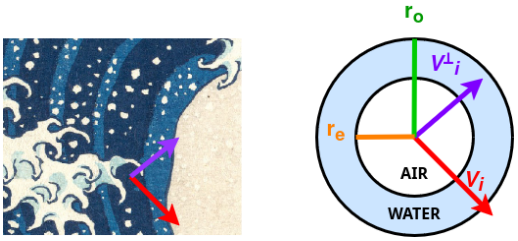


Figure 7: *To produce concave effects at wave crests (left), visual particles in $S_{concave}$ state distribute their air at the center (below r_e) and their water at the boundary (between r_e and r_o) of their volume (right). Their velocity v_i and perpendicular velocity $v_i^{\perp}$ are used to compute air entrainment.*

The concave shapes at the crest of waves typical of Japanese paintings are achieved using a specific distribution of air and water materials for visual particles in $S_{concave}$ state: the trapped air is at the center, and is surrounded by a thin liquid layer (see Figure 7). These two-material distributions are encoded using a single field function, negative near the center of the particle to represent the air carving out the wave crests, then positive between a given radius r_e and the full radius r_o of the particle, to represent the liquid envelope. Acting like an eraser, the negative part then produces rounded concavities when the contributions of all the particles are combined, while the positive part accentuates the peaks also observed at wave crests.

To ensure that the volume of liquid carried by a particle always remains constant, while the particle progressively entrains a given volume (or area in 2D) V_{air} of air as explained in Section 3.3, r_e

and r_o are computed as follows:

$$r_e = \begin{cases} RA_{\text{air}} \text{ if simulation is 2D} \\ RV_{\text{air}} \text{ if simulation is 3D} \end{cases}$$
$$r_o = \begin{cases} \left(R^2 + r_e^2\right)^{1/2} \text{ if simulation is 2D} \\ \left(R^3 + r_e^3\right)^{1/3} \text{ if simulation is 3D} \end{cases} \quad (5)$$

where we remind that R is the radius of the particle at rest. Depending on the dimensionality of the simulation, we use either air area A_{air} or air volume V_{air} of the particle.

To define the adequate negative-positive density field function from these values, we formulate the field as the concatenation of two cubic Hermite splines, as follows:

$$\begin{aligned} \phi(d) &= H_{00}(d)P_0 + H_{01}(d)P_0^{'} + H_{10}(d)P_1 + H_{11}(d)P_1^{'} \\ (P_o, P_1) &= \begin{cases} (\phi_{\min}, \phi_{\max}), & \text{if} \quad V_{\text{air}} > 0 \text{ for } d < r_m \\ (\phi_{\max}, 0), & \text{if} \quad V_{\text{air}} > 0 \text{ for } r_m \leq d \leq r_o \end{cases} \\ P_0^{'} &= 0 \\ P_1^{'} &= \begin{cases} \left(\frac{r_e}{r_o}\right)^{\frac{1}{2}}, & \text{if} \quad V_{\text{air}} > 0 \text{ for } r_m \leq d \leq r_o \\ 0 & \text{else} \end{cases} \end{aligned} \quad (6)$$

where the H terms are the standard Hermite spline scalar sub-functions, d is the distance of a point to the center of the visual particle, $r_m = r_e + (r_o - r_e)/2$ is the distance at which the density of water is maximal and ϕ_{max} is the value it takes there. Note that ϕ_{max} is always chosen as inferior to the iso-value (0.5 in our implementation), so that a single isolated particle in $S_{concave}$ state is not rendered as fluid. Instead, concave particles only influence and shape the surrounding liquid bodies by adding their field contributions to those of particles in $S_{droplet}$ state.

We add another mechanism to make the effect still closer to the reference art-piece: Noting that the curves peaks are not symmetric around a concavity, we introduce a local coordinate system aligned with the particle's velocity and adapt the density field to exert different influences depending on the angle relative to the motion. Let us consider a query point in a 2D space q, with $v_i^{\perp}$ such that $v_i \cdot v_i^{\perp} = 0$. We increase the effect of the density field function in the direction opposite to v_i as follows:

$$\begin{aligned} \hat{v}_i = \frac{\vec{v}_i}{|\vec{v}_i|}, \hat{v}_i^{\perp} = \frac{\vec{v}_i^{\perp}}{|\vec{v}_i^{\perp}|}, \hat{x}_{qi} = \frac{q - x_i}{|q - x_i|} \\ \Psi_{\vec{v}_i} = \text{clamp}\left(-\left(\hat{v}_i \cdot \hat{x}_{qi}\right) + \Psi_{\text{offset}}, 0, 1\right) \\ \Psi_{\vec{v}_i^{\perp}} = \max\left(\hat{v}_i^{\perp} \cdot \hat{x}_{qi}, 0\right) \\ \phi_{\max} = \phi_{\max} \Psi_{\vec{v}_i}^{1/8} \Psi_{\vec{v}^{\perp}}^{4} \end{aligned} \quad (7)$$

The scalar product between the particle's velocity $\vec{v}_i$ and the direction to the query point is used to check if the point lies in front of or behind the particle. Similarly, the scalar product with the vector perpendicular to the velocity, $\vec{v}_i^{\perp}$, is used to evaluate the vertical orientation relative to the particle. This reduces ϕ_{max} at query points in front of the particle in concave state and at lower positions.

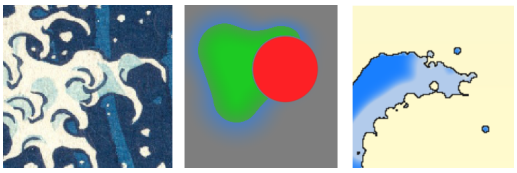


Figure 8: *2D results of the concave effect. From left to right: reference image; a visual particle in concave state (red, usually not depicted) carving a body of liquid formed by particles in droplet state (green); the concave effect applied at a larger scale.*

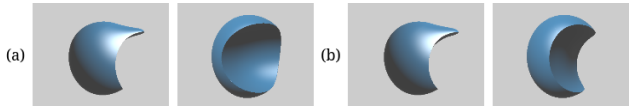


Figure 9: *Interaction between a 3D water droplet and a particle in concave state (not displayed but carving the first one) without (a) vs. with (b) a projected velocity over the camera plane. In the first sub-figures, the camera faces the plane where the two particles are located, requiring no correction.*

View-dependent effect. While the above method already generates the effects we seek for in the 2D case, as shown in Figure 8, some adaptations are needed in 3D. Indeed, the visual effect in Japanese paintings was designed for 2D illustrations, with concavities always showing their curved profiles. To obtain a visually similar effect on a 3D scene, the concavities should always be digged perpendicular to the view-point direction used for rendering the image. To achieve this, we use this direction to project the effect into the camera plane as shown in Figure 9. This is done as follows:

$$\begin{aligned} \vec{v}_i^{\perp} &= \vec{v}_i \times \vec{r}_d \\ \vec{v}_{\text{proj}} &= \vec{r}_d \times \vec{v}_i^{\perp} \\ \vec{v}_{\text{proj}}^{\perp} &= \vec{v}_{\text{proj}} \times \vec{r}_d \end{aligned} \quad (8)$$

where $\vec{r}_d$ is the ray from the camera and v_{proj} the projected velocity on the camera plane. $\vec{v}_{proj}$ and $\vec{v}_{proj}^{\perp}$ are then used in Equation (7) replacing $\vec{v}_i$ and $\vec{v}_i^{\perp}$.

4.3. Undersea Bubbles

A particle in S_{bubble} state share a similar air–water distribution with those in the $S_{concave}$ state, characterized by air enclosed within a liquid membrane. The main difference is that we increase the maximum positive field value ϕ_{max} beyond the 0.5-isovalue in bubble state, in order to produce the small white boundary as observed in Figure 2. We can also modify r_e and r_o computations to produce thinner or wider boundaries for the bubbles, using:

$$\begin{aligned} r_e^* &= r_e\left(1 - \frac{1}{4}\gamma\right) \\ r_o^* &= r_o\left(1 + \frac{1}{4}\gamma\right), \end{aligned} \quad (9)$$

where γ is the bubble width modifier and it can be controlled by the user, then r_e^* and r_o^* are used in (**??**).

Generating a streak. Another important visual effect that we observed in reference images from cartoons, is that the bubbles are often surrounded by a semi-transparent white region depicting the fact that there may be non-depicted, tiny bubbles around the main ones (see Figure 2 (c)). In our method, we model such regions, which we call streak, using an extra, larger SPH kernel W around each bubble particle. Its value W_q at a query point q is computed as:

$$\begin{aligned} \varepsilon &= \frac{(q - x_i)}{h} \\ W_{qi} &= \left(\frac{3.0}{\pi}\right) \exp(-\varepsilon^3) \\ W_q &= \sum_i V_{i,\text{air}} W_{qi}, \end{aligned} \tag{10}$$

where h is the influence of the SPH kernel, W_{qi} the contribution of particle i to q and W_q the overall contribution at q. The SPH kernel used here has a base circular shape but other shapes could be used as well (eg. elongated shapes in the direction of the bubbles motion) to obtain different streak shapes.

In the 3D case, bubbles can only be seen when the camera is underwater. In cartoons, they are either represented as white or as transparent spheres, often surrounded as a streak. We use white sphere in our implementation, and also provide the option to elongate the bubbles in their direction of motion, as follows:

$$\begin{aligned} |\vec{v}_i|^* &= \text{clamp}\left(|\vec{v}_i| - v^*, 0, 2\right) \\ r_o &= \left(r^\phi + r^\phi V_{i,\text{air}}\right)\left(1 + \frac{1}{2}\left(\vec{v}_i \cdot q_i\right)|\vec{v}_i|^*\right) \end{aligned} \tag{11}$$

where q is the field query point, x_i and v_i are the particle position and velocity, respectively, r_e is set to 0 to portray the white anisotropic bubbles, and a magnitude threshold coefficient v^* is used so that only bubbles with sufficient speed have an anisotropic shape. This mechanism allow us to represent in a more expressive way the velocity in a flow of bubbles.

4.4. Surface Foam

Foam in animation can be represented as bulky shapes like in Figure 2. Taking these as inspiration, we aim at representing both 2D and 3D foam as large, rounded shapes at, or slightly above, the surface of the liquid. The main challenge is that the visual particles in the S_{foam} state are not necessary above the surface due to the input simulation: they can either be exactly at surface level, or slightly below. To correct this, we apply a small vertical translation on the position of the visual counterparts of the simulated particles, to project the foam above the surface. To compute the bulky shapes, we set the r_e to 0 and r_o to:

$$r_o = \left(\frac{1}{\mu_i}\right)^2 (1 + \alpha_F V_{air}) r \tag{12}$$

Here we consider both the volume of air of the particle V_{air} and its density μ_p to increase its size. The value α_F is controlled by the user to change the foam scale. Lastly, as mentioned in Section 3.3, particles in S_{bubble} state may add air to the ones in S_{foam} state. Therefore the size of the foam bumps may increase over time.

5. Final Shape Generation and Rendering

To achieve the expressive rendering of the input simulation, a method for combining the visual effects we discussed remains to be defined. We first combine the field functions generated by the visual particles in the same state, and then use them to generate and render one or several implicit shapes, as detailed next.

5.1. Combining scalar fields

We compute the scalar field ϕ_S associated to state S by combining the individual scalar fields of the particle ϕ_p in this state, as follows:

$$\begin{aligned} \phi_{S_{\text{droplet}}} &= \sum \phi_{p \in S_{\text{droplet}}} \\ \phi_{S_{\text{concave}}} &= \overset{*}{\max}\left(\phi_{p \in S_{\text{concave}}}\right) \\ \phi_{S_{\text{bubble}}} &= \max\left(\phi_{p \in S_{\text{bubble}}}\right) \\ \phi_{S_{\text{foam}}} &= \max\left(\phi_{p \in S_{\text{foam}}}\right) \end{aligned} \tag{13}$$

Note that the summation of fields, resulting into a smooth implicit shape, is only used to blend particles in droplet state into a smooth water body. To get the desired sharp features where water bodies and concave particles interact, as well as in stylized bubbles and foam, we rather use sharp unions. This is implemented via a standard max for the bubble and foam, while we use an asymmetric operator $\max^*$ for the concave style, which takes the maximum of the positive contributions and the minimum of the negative ones, defined as follows:

$$\overset{*}{\max}\left(\phi_{p_1}, \phi_{p_2}\right) = \left\{ \begin{array}{cc} \phi_{p_1} & \text{if } |\phi_{p_1}| > |\phi_{p_2}| \\ \phi_{p_2} & \text{else} \end{array} \right\} \tag{14}$$

We then render one of several implicit shapes defined using a combinations of these fields as presented next.

5.2. Shape generation and Rendering

2D case. We use surface splatting to render different implicit shapes: One for water bodies, defined by the 0.5 iso-surface of $\phi_{S_{droplet}} + \phi_{S_{concave}}$, one for bubbles defined as the 0.5 iso-surface of $\phi_{S_{bubble}}$, and one for Foam, defined as the 0.5 iso-surface of $\phi_{S_{foam}}$. The three shapes are rendered with user-specified colors in different layers, with a salient contour line, and are finally superimposed. To improve visual rendering, we also render the bubbles streak, and optionally use the same mechanism to add a streak around concave and foam particles, if desired. Depending on the user choice, streak regions can be rendered either under or on top of the other layers, as semi-transparent surfaces with no contour line.

3D case. Rendering 3D animations is similar: If the camera is above the surface of the liquid, bubbles are discarded but we combine the other effects by rendering on one hand the 0.5 iso-surface of $\phi_{S_{droplet}} + \phi_{S_{concave}}$, and on the other hand the foam surface defined by the 0.5 iso-surface of $\phi_{S_{foam}}$. Note that the two surfaces may intersect. If the camera is under water, we add the rendering of the bubbles as semi-transparent objects, using the 0.5 iso-surface of $\phi_{S_{bubble}}$. In our implementation, we used ray-casting for rendering.

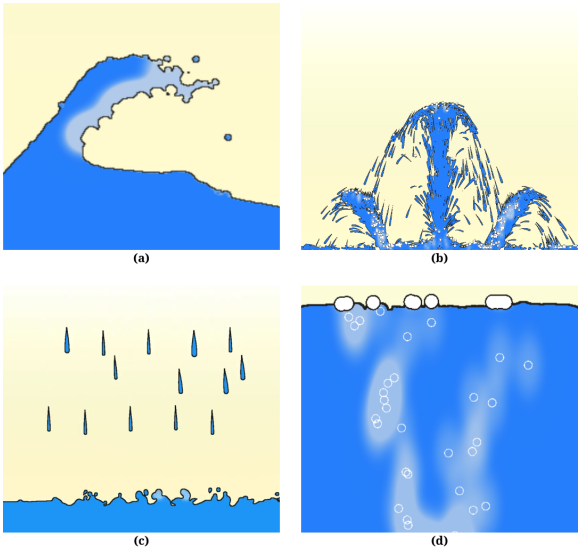


Figure 10: *Collection of 2D visualizations obtained with our method, here we show: (a) Crest of wave, (b) Fountain, (c) Rain droplets and (d) Column of bubbles going to surface.*

6. Results and Discussions

6.1. Implementation and Performances

We used the tool SPlisHSPlasH [B*16] to run the simulations and extract the relevant data for the tests. The state machine transitions for all particles are computed with parallel computing using multi-threaded CPU, and the rendering stage is implemented with OpenGL + CUDA in C++. Rendering was computed on a GPU NVIDIA RTX 4070, with uniform grids used as acceleration structures both in the 2D and 3D cases. The average time to render a frame of a 2D animation with more than 20K particles was 0.3 seconds, while in the 3D case, scenes took 2.5 seconds, 48 seconds, and 75 seconds on average to render, corresponding to 10K, 88K, and 160K particles, respectively.

6.2. 2D Results

Figure 10 shows the results of different 2D animation scenarios. Thanks to the effect state-machine, our solution enables the progressive formation of temporally coherent cavities at the crest of waves (see Figure 10 (a)), which only disappear when the wave collapses and concave particles switch to the bubble state (see Figure 19). In this first example only $S_{droplet}$ and $S_{concave}$ were activated to make the result similar to the Japanese painting reference, but we used a streak generated by concave particles to add semi-transparent white colors over the crest of the wave. In the fountain example (see Figure 10 (b) and Figure 18), the use of the expressive droplets achieves cartoon-like rendering, and parameters were set to generate curved tails that resemble the ones observed in Figure 2. The same model was also applied to depict a rainy scene, illustrated in Figure 10 (c) and Figure 16. Finally Figures 10 (d) and 17 show a column of rising bubbles transforming into foam, inspired by Figure 2 (c) and (d). Note how the foam is kept at surface level, and its blending is adapted to look like the one in the reference picture. In addition, to illustrate again the amount of user control, Figure 14 shows different renderings of the same bubbles scene obtained by editing the shape of the streak.

6.3. 3D Results

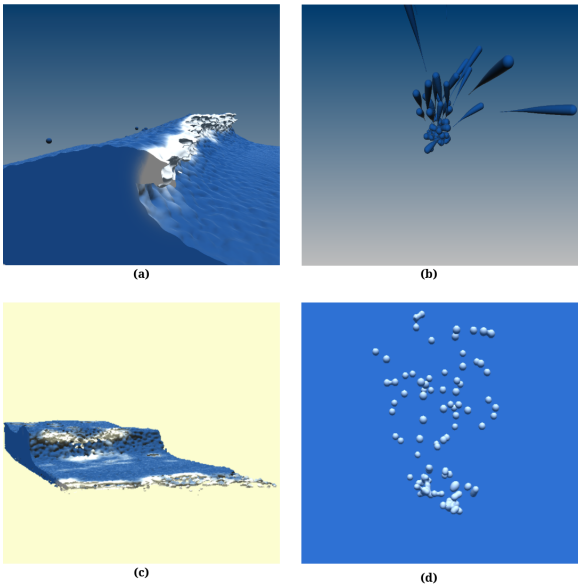


Figure 11: *Display of different 3D visualization obtained with our method. From left to right we have: (a) Crest of a big wave, (b) Droplets falling, (c) Offshore small wave, (d) Column of bubbles.*

Some of our renderings of 3D animations are gathered in Figure 11: (a) shows the same wave crest than in Figure 1, but from a different viewing angle, showing 3D concave effect remains salient whatever the camera viewpoint, thanks to the adaptation. Figure 21 shows the effect of the resolution at which the simulation was computed on the rendered effect. While increasing the number of simulated particles can produce a greater number of smaller concavities, the sharpness of the visual features, such as droplet tails and concavities, is independent of particle resolution. As a result, even a coarse simulation can be transformed into an expressive one with sharp visual details using our method. Finally, we show in Figures 11(d) and 20 an example of underwater scenario where a columns of air is modeled thanks to the advection of the air component of visual particles in bubble state.

6.4. Perceptual validation

To assess the perceptual quality of our results, we conducted a user study comparing 2D and 3D animations produced with our approach against those generated by state-of-the-art AI tools available to the public such as Firefly [Ado] and DomoAI [Dom]. Specifically, we evaluated the hypothesis H_0: *Our method better captures the style of the reference cartoon images than an LLM-based generative AI tool* in both the 2D and 3D cases.

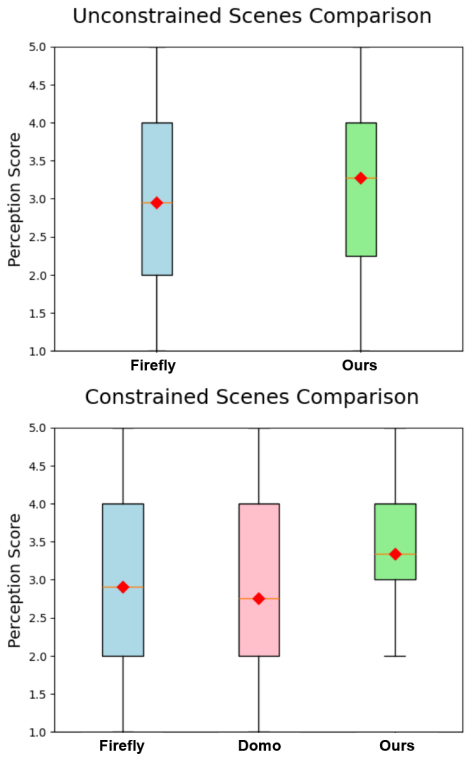


Figure 12: *User study results for Unconstrained scenes (top) versus constrained scenes were the AI was given our input particle-based animation clip for extra conditioning (bottom). Higher score means better, the mean being represented by the red diamond.*

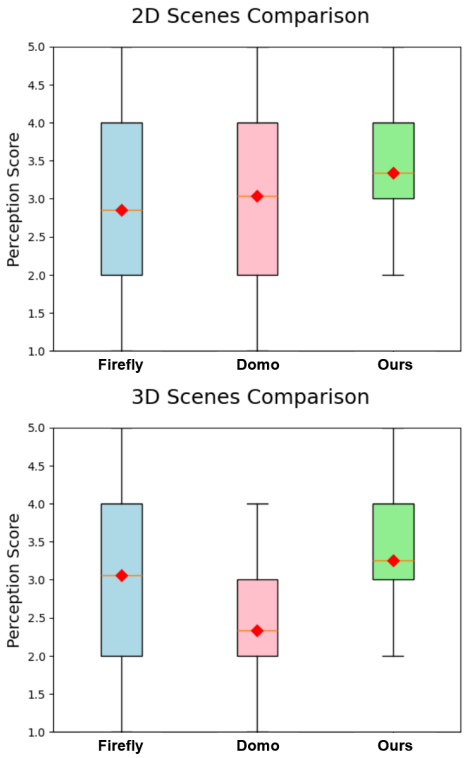


Figure 13: *The same results are analyzed according to the 2D or 3D nature of the scene. Higher means better, the mean being represented by the red diamond.*

#	Scene	Firefly	Domo	Ours
1	Fountain 2D	2.84	-	3.48
2	Fountain 3D	3.55	-	3.06
3	Wave 2D	2.84	-	3.16
4	Wave 2D	4.00	-	3.06
5	Wave 3D	2.71	-	3.35
6	Bubbles 2D	1.74	-	3.55
7	Fountain 2D	3.13	2.48	3.42
8	Fountain 3D	3.45	2.23	3.16
9	Wave 2D	2.06	4.06	3.06
10	Wave 3D	2.55	2.45	3.42
11	Bubbles 2D	3.35	2.55	3.61

Table 1: *User study perception score per question, with from top to bottom Unconstrained (Q1-6) and then Constrained (Q7-11) comparisons. The worst (red) and best (green) scores are highlighted in each segment for every method.*

Task Participants were asked to perform a perceptual evaluation of videos based on a provided description. After reading the description, videos produced with both a generative-AI tool and our method were shown, in random order. Participants rated the correspondence between the description and each video on a scale from 1 to 5, where 1 indicates low correspondence and 5 indicates high correspondence. They could re-watch or pause the videos as often as they wished and were free to answer the questions in any order.

Protocol Before going through the study, participants were asked to provide their age, gender, and expertise level in visual art and computer graphics. All participants answered the same set of questions and were shown the same videos. The survey was divided into two types of comparisons, unconstrained scenes (US) and constrained scenes (CS), where the input particle-based simulation was used for further conditioning the AI generative models. For the US, we compared against Firefly [Ado]. We provided a detailed prompt, the tool being free to generate both motion and style from the prompt. For the CS comparison, we compared against both Domo [Dom] and Firefly, removing any detailed motion description from the prompt but providing the video of particles motion. The same particle simulation results were used to produce our own stylized animations. The participants were not informed of what method or tool was used for each video, nor about the US and CS nature of the comparison.

Participants The survey was distributed to a diverse set of participants through various online platforms and in multiple languages. A total of 31 participants completed the survey, ranging in age from 18 to 59 years (21 male, 9 female, and 1 unspecified). Among them, 77% reported having an intermediate to advanced knowledge in computer graphics research or graphical art.

Analysis Following the results of the survey, we performed a Wilcoxon signed rank test with our hypothesis as H_1 to evaluate if on average our results were better. We conduct the test separately in the UC and SC cases. A summary of the survey results can be observed in Table 1, Figure 12 and Figure 13).

Following our Wilcoxon test, our results were: US scenes $Ours - Firefly(statistic = 5052, p < 0.001)$ and, in CS scenes $Ours - Firefly(statistic = 3124.5, p < 0.001)$ and $Ours - Domo(statistic = 7305.5, p < 0.001)$. This indicates that in US our method performs better than Firefly, while in CS our method also performs better than both Firefly and Domo. Taking into account the Wilcoxon test results, we confirm that our method indeed captures better the style of the references images in the chosen scenarios. The results in Figure 13 shows the difference between means of our method and the LLM-based tools. We also observed that the AI tools achieved the highest and lowest scores observed. For the 2D animation clips, we obtained an average of 3.34 of perception score and an average difference of 0.4 compared to the tools, our best perceived result being Bubbles 2D, an underwater column of bubbles rising to the surface where the AI struggled in particular with the air-water interaction at the surface level. On the other hand, AI tools managed to portray other scenes much better like the Wave 2D, a collapsing wave. This scene is an interesting case where, despite not having an approach based on geometry stylization, the AI tools achieved better perceived 2D results thanks to style transfer and to the visual elements they added based on the prompt like foam. In contrast, our own result was presented in a very basic rendering style. The AI struggled to properly extend the effects observed in 2D cartoons to 3D, while this was a strength of our method. An example of this is the Wave 3D where we mimic the 2D shapes observed in the Japanese big wave. Overall, the results produced by our method were evaluated as more convincing than those of the AI tools in the unconstrained and constrained cases, and for both 2D and 3D scenes. We are however aware that, with a more accurate description and improved style configuration, an AI tool could become quite competitive and – at the price of many trials and errors – probably be used to produce similar or better results than ours.

6.5. Limitations

Although our expressive rendering pipeline achieves the desired effects in both 2D and 3D, thanks to the versatility of implicit modeling, we observed a number of limitations:

First, we sometimes observed consistency issues resulting in visual artifacts, when chaotic dynamic motion produces very quick changes of state for several neighboring particles. To avoid any visual artifact, we added a minimum lifetime for particles in each effect state, ensuring a smoother transition between effects. Second, the thin tails of the expressive droplets could not be rendered using our grid-based acceleration structure, due to sharp cuts or blinking when using the structure. This resulted in an increased rendering time for some of our scenes. Third, regarding bubbles and foam, some bubbles may appear stuck in place when particles in bubble state continuously transfer air to the same stationary neighbor. A possible solution would be to introduce a small random variation in the air flow direction used in Equation (1), allowing the target particle to change over time. Conversely, bubble state transfer can create too jerky a motion if the particles are large. This could be solved by a gradual air transfer, tuned to create a bubble in an intermediate position between the source particle and its neighbour. Fourth, although our method does not require high-resolution simulations to produce sharp and expressive features, the resolution at which our visual effects are generated remains directly tied to the resolution of the input simulation. Still, we may want to render an animation precomputed with millions of particles while applying stylized concavities, droplets, and blobby foam at a much coarser scale. In this case, particles should be resampled before our rendering is applied, which is left for future work. Lastly, while our method can achieve more convincing results than the AI tools, the latter can still switch easily to different artistic styles so are much more widely applicable than our current method. While the identified effects might be enough to produce stylized animations, our procedural approach requires to consider the different styles that these effects could have, and to develop dedicated methods.

7. Conclusions and Future Work

We proposed a method for generating expressive renderings of liquid animations, inspired by 2D artworks and cartoons. We extract information from a particle-based simulation to create a visual counterpart to each simulated particle, driven by an effect state machine to portray a set of pre-identified visual effects. Based on field functions generating implicit contours or surfaces, our solution captures the visually salient sharp features present in art-works. Moreover, it is versatile and can be easily applied to both 2D and 3D input animations.

In future work, we plan to improve the rendering of some of the effects, such as the behavior of bubble reaching the surface level, by extending recent models [ALH*20,MRR21] to bubbles and foam in a cartoon style. Moreover, while we limited our state machine to a set of four predefined stylization effects (namely droplets, concave shapes, bubble and foam), our framework could be easily extended to other artistic styles. We could even investigate the direct generation and control of a new effect from a user's sketch, inspiring from recent sketch-based solutions in implicit modeling [ATW*17], this can extend even more the range of artistic styles that our method could handle.

References

[ACCK18] ANG N., CATLING A., CIARDI F. C., KOZIN V.: The technical art of sea of thieves. In *ACM SIGGRAPH 2018 Talks* (New York, NY, USA, 2018), SIGGRAPH '18, ACM. 2

[Ado] ADOBE: Adobe. (2025). Adobe Firefly, Image Model 4 [Generative AI model]. Adobe Systems Incorporated. https://www.adobe.com/products/firefly.html. 8, 9

[ALH*20] ATASI O., LEGENDRE D., HAUT B., ZENIT R., SCHEID B.: Lifetime of surface bubbles in surfactant solutions. *Langmuir 36*, 27 (2020), 7749–7764. 10

[ATW*17] ANGLES B., TARINI M., WYVILL B., BARTHE L., TAGLIASACCHI A.: Sketch-based implicit blending. *ACM Trans. Graph. 36*, 6 (2017). 3, 10

[B*16] BENDER J., ET AL.: SPlisHSPlasH Library, 2016. 8

[BBB*97] BLOOMENTHAL J., BAJAJ C., BLINN J., CANI M.-P., ROCKWOOD A., WYVILL B., WYVILL G.: *Introduction to Implicit Surfaces*. Morgan Kaufmann, 1997. 3

[BBRF14] BROWNING M., BARNES C., RITTER S., FINKELSTEIN A.: Stylized keyframe animation of fluid simulations. In *Proceedings of the Workshop on Non-Photorealistic Animation and Rendering* (2014), NPAR '14, ACM. 2

[BKKW19] BENDER J., KOSCHIER D., KUGELSTADT T., WEILER M.: Turbulent micropolar sph fluids with foam. *IEEE Transactions on Visualization and Computer Graphics 25*, 6 (2019), 2284–2295. 2, 4

[Bri15] BRIDSON R.: *Fluid Simulation for Computer Graphics (2nd ed.)*. A K Peters/CRC Press, 2015. 2

[BUH15] BAEK S., UM K., HAN J.: Muddy water animation with different details. *Computer Animation and Virtual Worlds 26*, 3-4 (2015), 347–355. 2

[CD97] CANI M.-P., DESBRUN M.: Animation of Deformable Models Using Implicit Surfaces. *IEEE Transactions on Visualization and Computer Graphics 3*, 1 (1997), 39 – 50. Published under the name Marie-Paule Cani-Gascuel. 3

[CGB16] CANEZIN F., GUENNEBAUD G., BARTHE L.: Topology-aware neighborhoods for point-based simulation and reconstruction. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2016), SCA '16, p. 37–47. 3

[CSGC16] CORDIER F., SINGH K., GINGOLD Y., CANI M.-P.: Sketch-based Modeling. In *Siggraph Asia 2016* (Macao, China, 2016), SIGGRAPH ASIA 2016 Courses, ACM, p. 222. 3

[CSS*25] CHEN Y., SHAO G., SHUM K. C., HUA B.-S., YEUNG S.-K.: Advances in 3d neural stylization: A survey. *International Journal of Computer Vision 133*, 8 (Aug 2025), 5026–5061. URL: https://doi.org/10.1007/s11263-025-02403-9, doi:10.1007/s11263-025-02403-9. 3

[DHV*23] DROSKE M., HANIKA J., VORBA J., WEIDLICH A., SABBADIN M.: Path tracing in production: The path of water. In *ACM SIGGRAPH 2023 Courses* (New York, NY, USA, 2023), SIGGRAPH '23, ACM. 2

[Dom] DOMOAI: DomoAI: Video-to-video, style transfer, AI, video tool. https://www.domoai.app/. 8, 9

[DYZ*23] DENG Y., YU H.-X., ZHANG D., WU J., ZHU B.: Fluid simulation on neural flow maps. *ACM Trans. Graph. 42*, 6 (Dec. 2023). URL: https://doi.org/10.1145/3618392, doi:10.1145/3618392. 2

[EBG*07] EDEN A. M., BARGTEIL A. W., GOKTEKIN T. G., EISINGER S. B., O'BRIEN J. F.: A method for cartoon-style rendering of liquid animations. In *Proceedings of Graphics Interface 2007* (New York, NY, USA, 2007), ACM. 2

[GBC*13] GOURMEL O., BARTHE L., CANI M.-P., WYVILL B., BERNHARDT A., PAULIN M., GRASBERGER H.: A Gradient-Based Implicit Blend. *ACM Transactions on Graphics 32*, 2 (2013). 3

[IAAT12] IHMSEN M., AKINCI N., AKINCI G., TESCHNER M.: Unified spray, foam and air bubbles for particle-based fluids. *Vis. Comput. 28*, 6-8 (jun 2012), 669–677. 2, 4

[IBAT11] IHMSEN M., BADER J., AKINCI G., TESCHNER M.: Animation of air bubbles with sph. In *International Conference on Computer Graphics Theory and Applications* (2011). 2

[JFA*15] JAMRIŠKA O., FIŠER J., ASENTE P., LU J., SHECHTMAN E., SÝKORA D.: Lazyfluids: appearance transfer for fluid animations. *ACM Trans. Graph. 34*, 4 (July 2015). URL: https://doi.org/10.1145/2766983, doi:10.1145/2766983. 3

[KAGS20] KIM B., AZEVEDO V. C., GROSS M., SOLENTHALER B.: Lagrangian neural style transfer for fluids. *ACM Trans. Graph. 39*, 4 (2020). 2

[KAOT23] KANYUK P., AZEVEDO V., ORTIZ R., TANG J.: Singed silhouettes and feed forward flames: Volumetric neural style transfer for expressive fire simulation. In *ACM SIGGRAPH 2023 Talks* (New York, NY, USA, 2023), SIGGRAPH '23, Association for Computing Machinery. URL: https://doi.org/10.1145/3587421.3595435, doi:10.1145/3587421.3595435. 2

[KCG*14] KAZI R. H., CHEVALIER F., GROSSMAN T., ZHAO S., FITZMAURICE G.: Draco: bringing life to illustrations with kinetic textures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2014), CHI '14, Association for Computing Machinery, p. 351–360. URL: https://doi.org/10.1145/2556288.2556987, doi:10.1145/2556288.2556987. 3

[LKLK19] LEE J., KIM J.-H., LEE H.-Y., KIM S.-J.: Realistic fluid representation by anisotropic particle. *J. of Visualization 22*, 2 (2019). 2

[LYP11] LIAO J., YU J., PATTERSON J.: Modeling ocean waves and interaction between objects and ocean water for cartoon animation. *Computer Animation and Virtual Worlds 22*, 2-3 (2011), 81–89. 2

[LZ13] LIU S., ZHANG W.: Cartoon-style simulation of water coupled with objects. *Sim. Modelling Practice and Theory 31* (2013). 2

[MRR21] MIGUET J., ROUYER F., RIO E.: The life of a surface bubble. *Molecules 26*, 5 (2021). 10

[MWK22] MA J., WEI L.-Y., KAZI R. H.: A layered authoring tool for stylized 3d animations. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2022), CHI '22, Association for Computing Machinery. URL: https://doi.org/10.1145/3491102.3501894, doi:10.1145/3491102.3501894. 3

[OBG*24] OLIVIER P., BUTLER T., GUEHL P., COLL J.-L., CHABRIER R., MEMARI P., CANI M.-P.: DynBioSketch: A tool for sketching dynamic visual summaries in biology, and its application to infection phenomena. *Computers and Graphics 122* (Aug. 2024), 103956. URL: https://hal.science/hal-04682753, doi:10.1016/j.cag.2024.103956. 3

[OC97] OPALACH A., CANI M.-P.: Local Deformation for Animation of Implicit Surfaces. In *Spring Conference on Computer Graphics (SCCG)* (Bratislava, Slovakia, 1997), Straßer W., (Ed.), pp. –. Published under the name Marie-Paule Cani-Gascuel. 3

[RNA14] ROCHA NETO L. D. S., APOLINARIO A. L.: Cartoon water rendering with foam and surface smoothing. In *2014 Brazilian Symposium on Computer Games and Digital Entertainment* (2014). 2

[TKAS23] TANG J., KIM B., AZEVEDO V. C., SOLENTHALER B.: Physics-informed neural corrector for deformation-based fluid control. *Computer Graphics Forum 42*, 2 (2023), 161–173. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14751, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14751, doi:https://doi.org/10.1111/cgf.14751. 2

[WFS22] WRETBORN J., FLYNN S., STOMAKHIN A.: Guided bubbles and wet foam for realistic whitewater simulation. *ACM Trans. Graph. 41*, 4 (jul 2022). 2, 4

[WSB25] WRETBORN J., STOMAKHIN A., BATTY C.: A unified multiscale method for simulating immersed bubbles. *Computer Graphics Forum* (2025). 2

[XASM24] XIE H., ARIHARA K., SATO S., MIYATA K.: Dualsmoke: Sketch-based smoke illustration design with two-stage generative model. *Computational Visual Media 10*, 5 (Oct 2024), 965–979. URL: https://doi.org/10.1007/s41095-022-0318-0, doi:10.1007/s41095-022-0318-0. 3

[XKG*16] XING J., KAZI R. H., GROSSMAN T., WEI L.-Y., STAM J., FITZMAURICE G.: Energy-brushes: Interactive tools for illustrating stylized elemental dynamics. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (New York, NY, USA, 2016), UIST '16, Association for Computing Machinery, p. 755–766. URL: https://doi.org/10.1145/2984511.2984585, doi:10.1145/2984511.2984585. 3

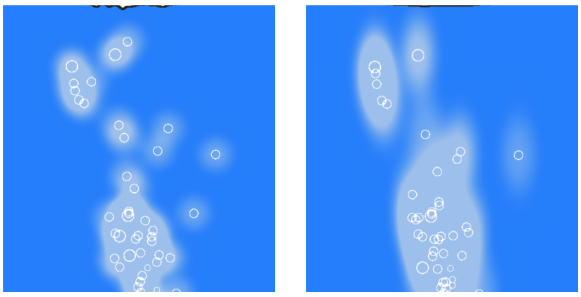


Figure 14: *Tuning the bubbles streak. Comparison between a circular (left) and an ellipsoidal kernel oriented in the velocity direction of each particle in S_{bubble} state (right).*

Figure 15: *2D animation of a splash. Note that a droplet's tail follows its trajectory and curves when the trajectory does. Its length reflects the droplet's speed.*

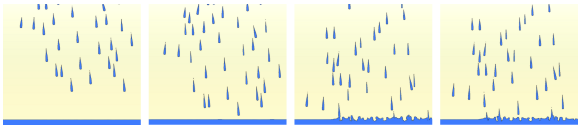


Figure 16: *2D animation of rain splashing on a water surface.*

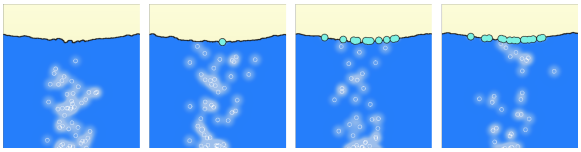


Figure 17: *A 2D under-water air column. Bubbles propagate upward even if particles don't move, thanks to the propagation of bubble state. Foam appear when bubbles reach the surface.*

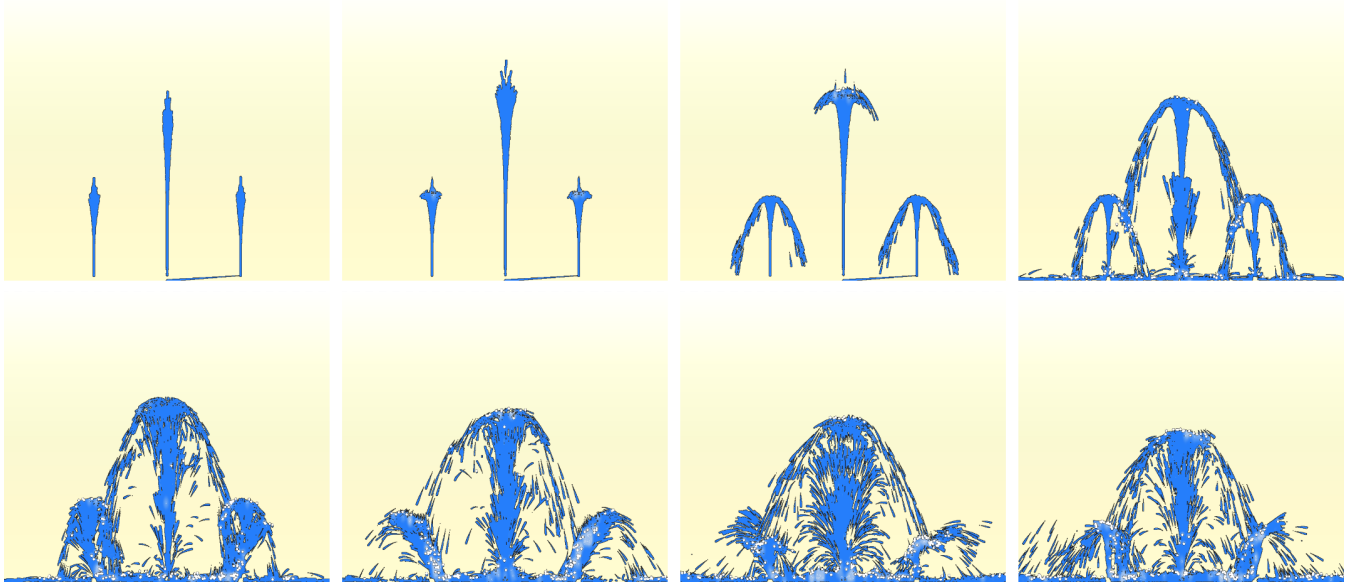


Figure 18: *Sequence of a fountain 2D animation (left to right, top to bottom) showing the mixture of effects from our method over an interval between the start of the fluid emission and the mixture of columns of fluids.*

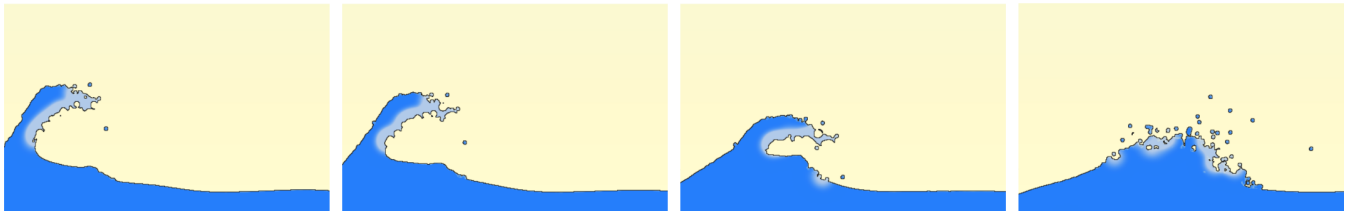


Figure 19: *2D animation of the collapse of a wave, simulated using* 26*K particles. Particles in concave state at the crest of the wave are used both to dig cavities and to generate a white streak opposite to their velocity direction, to emphasize turbulent motion at the crest. Here, the elongated droplets effect was disabled, since circular droplets better resemble the art-piece we inspired from (shown in Figure 1, left).*

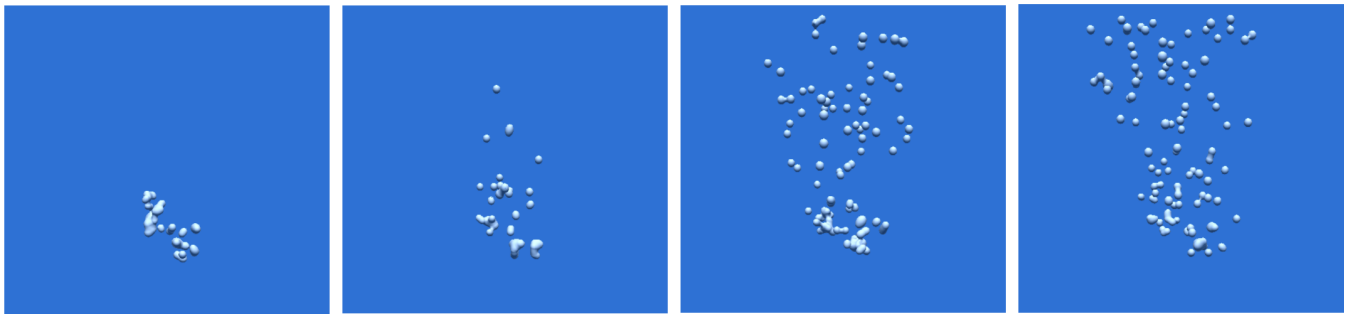


Figure 20: *A 3D underwater scene with a column of bubbles modeled with our method.*

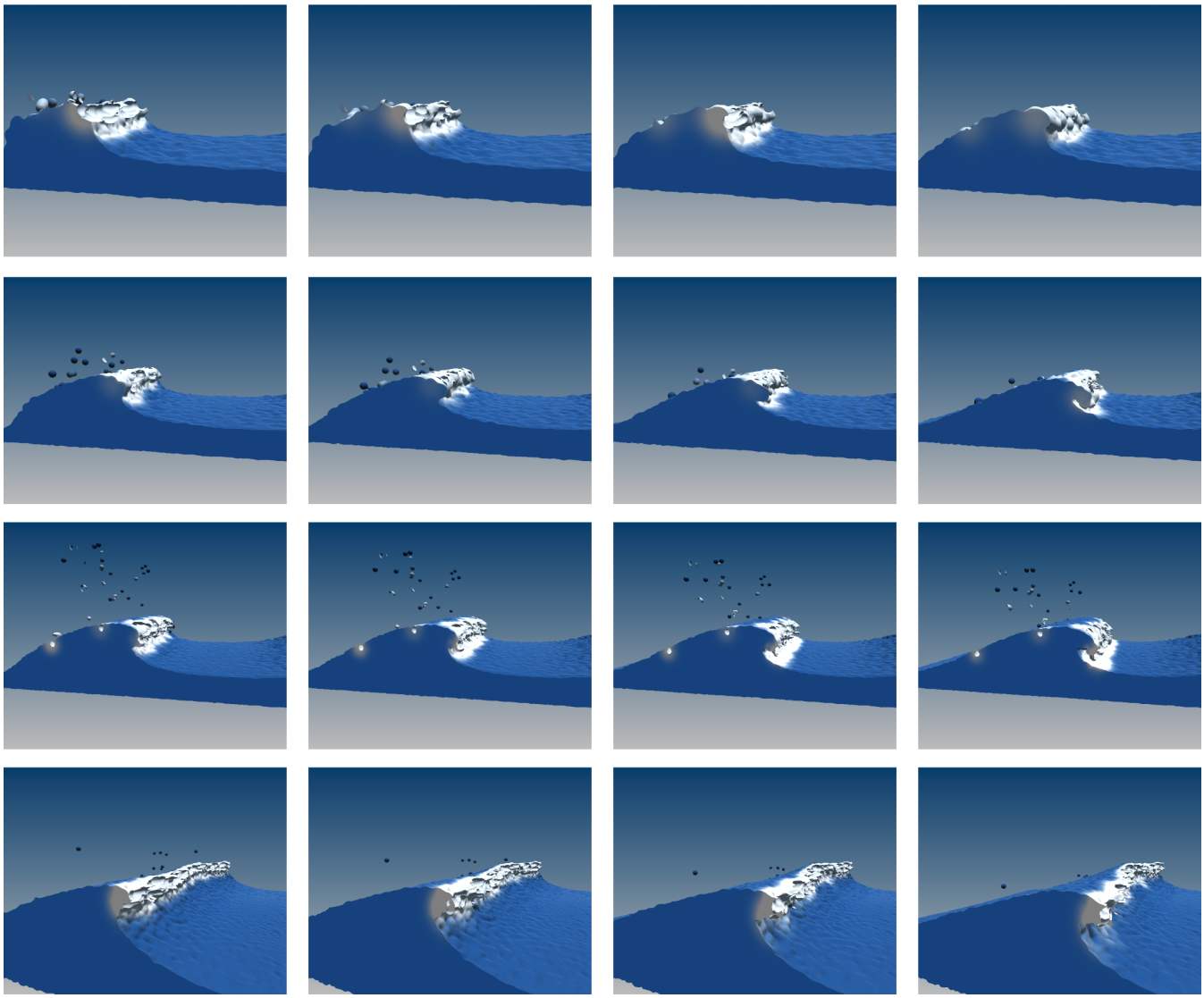


Figure 21: *Another wave sequence, illustrating the moment between the climax of the wave and the beginning of the collapse. As shown from top to bottom, the spatial resolution of the simulation plays a strong role in the visual results: We respectively used* $\sim 10K$ *particles (top),* $\sim 24K$ *particles (second line),* $\sim 88K$ *particles (third line) and* $\sim 170K$ *particles (bottom).*