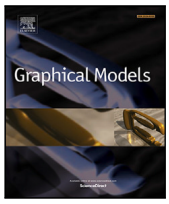
Graphical Models 147 (2026) 101340

Contents lists available at ScienceDirect

Graphical Models

journal homepage: www.elsevier.com/locate/gmod



Stylizing 2D & 3D liquid animations through adaptive particle-based implicit fields☆

Rodrigo Stevenson-Regla [a,*], Damien Rohmer [a], Loïc Barthe [b], Marie-Paule Cani [a]

[a] LIX, École Polytechnique, CNRS, IP Paris, Paris, France
[b] IRIT, Université Toulouse III, CNRS, Toulouse, France

ARTICLE INFO

Keywords:
Implicit surfaces
Non-photorealistic rendering
Animation

ABSTRACT

Combining physically-based simulation with stylized visual effects not only requires changing the appearance 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 kinematic information from the simulation. These density fields are combined at each frame to create the implicit iso-surface of interest, rendered in an adapted style. By defining a 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 , which we further enhance in 3D scenes using a dedicated stylized surface-color pattern. 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. 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 and a hands-on evaluation with a professional artist, in a variety of animation scenarios where stylized shapes are needed. This is an extension of our previous work presented at STAG 2025. It introduces new material, including a refined blending operator, a stylized surface-color pattern, performance metrics, and artist evaluations.

1. Introduction

Water animations have been used for many years in movies, cartoons, video games, and scientific visualizations. While most research work focused on the modeling of the realistic behavior of liquids, stylized cartoons and paintings rather depict liquids using exaggerated representations typically featuring elongated droplet shapes, or bulks associated with foamy surfaces (see Fig. 2). These details correspond to changes in the 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 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 at a constant value. The surface to display is obtained as an isosurface of the global field formed by summing all individual particle contributions, 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

☆ This article is part of a Special issue entitled: 'YGMOD_STAG 2025' published in Graphical Models.

* Corresponding author.
E-mail addresses: stevenson@lix.polytechnique.fr (R. Stevenson-Regla), damien.rohmer@polytechnique.edu (D. Rohmer), loic.barthe@irit.fr (L. Barthe), marie-paule.cani@polytechnique.edu (M.-P. Cani).

https://doi.org/10.1016/j.gmod.2026.101340
Received 29 April 2026; Received in revised form 23 July 2026; Accepted 4 August 2026
Available online 17 August 2026

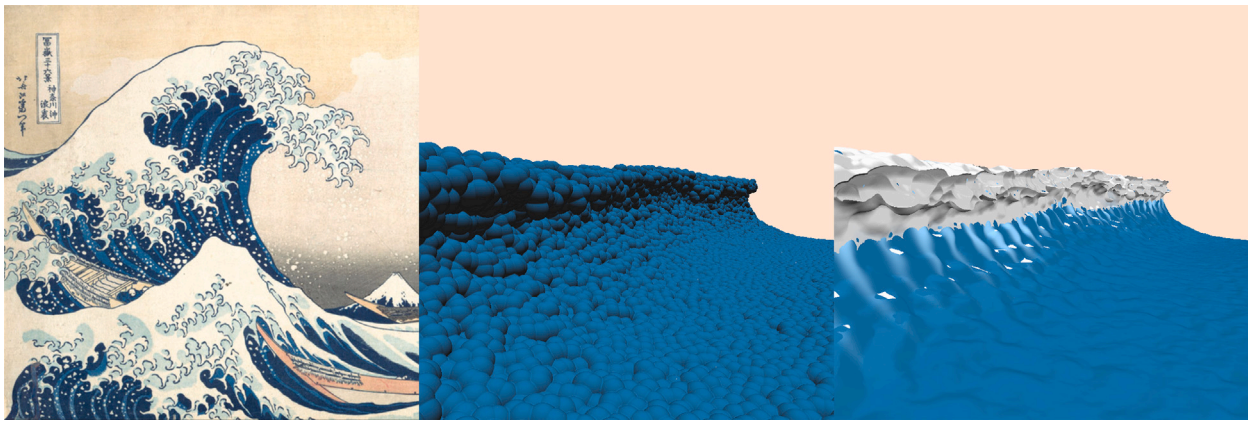


Fig. 1. Our method stylizes a standard 3D particle-based simulation to match typical artwork effects. Left: Reference artwork; Middle: Raw 3D particle-based simulation; Right: Stylized result obtained by applying our method, exhibiting geometrical deformations (carved concavities at the tip of the wave) and color stylization effects driven by the underlying simulation. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Fig. 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. Please note that figures (a), (c) and (d) were generated with Nano Banana [1] to mimic the visual style of original artworks from Tom & Jerry and Pinocchio. The original frames are not presented here to respect copyright protections. Image (b), Katsushika Hokusai's The Great Wave off Kanagawa, is in the public domain and is presented unmodified. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

by a state machine that triggers the appearance of and transitions 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; this includes a blending scheme for our generated surfaces towards the target effects and a stylized surface-color pattern, illustrated on breaking-wave scenes, that further enriches the rendering by exploiting the information provided by our visual particles.

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-photorealistic rendering, our tool acts as a bridge between particle computations and stylized visualizations. An illustration, showing the reference artwork, the raw particle simulation, and the stylized 3D result of the tool, is provided in Fig. 1. We validate our approach with a perceptual user study comparing it to state-of-the-art generative-AI tools, and report a hands-on case study with a professional 2D artist where we analyze its limitations and ease of use in different scenes. A previous version of this work was presented at STAG 2025 [2]. This journal version substantially extends the original framework by introducing new material. The primary increments in contribution include a refined asymmetric blending operator for stylized air–water mixtures (Section 5.1), a spatially varying surface-color pattern for breaking waves (Section 5.3), a detailed performance benchmark (Section 6.1), and a qualitative case study with a professional artist (Section 6.6).

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 [3,4]. 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 the modeling of air entrapment as a second step (see [5] for a full review). Ihmsen et al. [6] defined a criterion with three characteristics to measure air entrapment. Bender et al. [7] improved this by adding vorticity as a fourth characteristic. Wretborn

et al. [8] proposed an aeration criterion based on particle stress. Baek et al. [9] showed the interaction between solids and liquid for muddy water. Ihmsen et al. [10] elaborated a method to animate air bubbles with SPH methods. Wretborn et al. [11] proposed a method for diffuse bubbles for different models of bubble distributions.

While the methods above aimed at photorealistic rendering of simulated liquids, artworks and cartoons introduced a variety of stylized water representations, which often differ greatly from real phenomena, see Fig. 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. [12] presented a method for rendering water based on cartoon animations. Liao et al. [13] depicted ocean waves based on drawn animation styles. Liu and Zhang [14] used elliptical shapes to model splashes when interacting with ships or rocks. Neto and Apolinario [15] 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. [16] introduced a method that uses keyframe generation to portray hand-drawn fluids. Lee [17] used 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. [18], 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 [19–21] achieved both changes in visual style and geometric shape, but mostly focused on fire and smoke. These methods are therefore not applicable to water, or water and air mixtures.

In summary, as shown by a recent survey [22], stylization of liquids remains a challenging task both in terms of availability of the training databases and user control over the desired effects.

In this work, we build on the versatility of implicit modeling [23], a perfect representation for animating highly deformable shapes that split and merge over time [24–26].

Another source of inspiration is sketch-based animation [27,28], where advanced blending operators such as those proposed by Gourmel et al. [29] 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. [30] generated animations by adding coordinated motions to collections of objects in illustrations. Similarly, Jamriška et al. [31] introduced a method to create sprites from static images through flow-guided texture synthesis. Xing et al. [32] set up an interface to create stylized effects based on user-controlled velocity fields. Ma et al. [33] 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 animating cells carried by a fluid [34]. Lastly, Xie et al. [35] proposed a sketch-based interface building on Lagrangian style transfer to create smoke illustrations.

3. Visual particles and state changes

This section describes the overall architecture of our pipeline and its key components, which generate stylized effects from a particle-based simulation as a post-processing step.

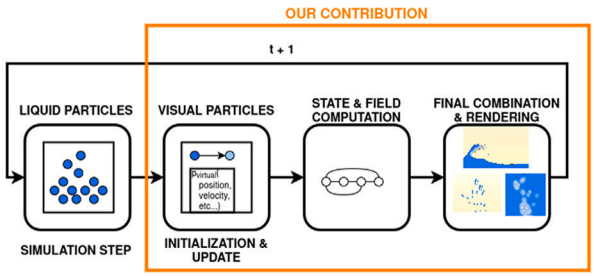


Fig. 3. Processing pipeline of our method, where our contribution is highlighted. After each simulation step (left) we either initialize (first step) or update each visual particle, compute their effect-state and the associated air and water density fields (middle), and finally combine their contributions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

3.1. 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 in a similar change for their visual counterpart. Conversely, 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 only the positions 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 end, the visual 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.

3.2. Processing pipeline

Our processing pipeline is presented in Fig. 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 particles' scalar fields in order to generate the final, stylized image.

This pipeline applies regardless of the simulation's dimensionality: it can render 2D simulations in a flat cartoon-like style , or produce stylized renderings of 3D animations (Fig. 1, right), with only minimal adaptations to the effects. The 2D and 3D versions of each effect are discussed in Section 5.

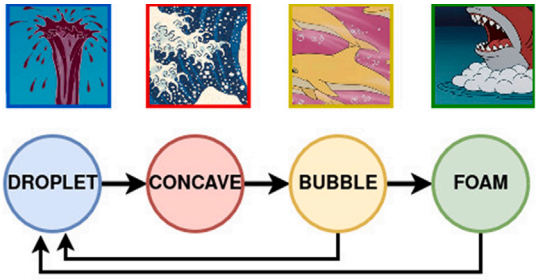


Fig. 4. The effects state machine. A particle can only transition from a state to one of its successors in this oriented graph. Each state encodes a given visual effect, and each oriented edge comes with a set of conditions for applying the transition.

3.3. Changes of state

The visual effects produced by the visual 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 Fig. 2. Our state machine, depicted in Fig. 4, therefore has four states: the **droplet state** S_D used for water-only particles, the **concave state** S_C targeting stylized concave regions at the crest of breaking waves, the **bubble state** S_B used for undersea bubbles, and finally the stylized **foam state** S_F. 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_D, used to represent both water bodies and expressive droplets. A particle may then immediately switch to state S_C if it is detected to be at the crest of a wave.

Particles are considered to belong to a wave crest when the liquid surface they contribute to exhibits high surface curvature. We use the wave crest detection method defined in [6] 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_D to S_C.

Once in state S_C, 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 criterion defined in [6]. 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 5).

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 a change, our implementation follows the approach of Ihmsen et al. [6], 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_C to S_B when the difference in velocities between the particle and its neighbors is below a given threshold.

Once in state S_B, 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_D (as it no longer contains air), while the neighboring particle switches to S_B. 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 [7] to model such underwater air flow, while also providing two parameters under the user's control. This is formulated as follows:

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

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 $\tilde{v}_i$ 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_B reaches the surface, it switches to state S_F. 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_F does not transfer the air it contains anymore, but it can still receive air from other particles at state S_B. Particles at state S_F have a minimum lifetime and, when it is reached, they can burst, releasing all their air and going back to S_D. This process is inspired by the lifetime defined in [8] where this lifetime is related to the amount of neighboring foam.

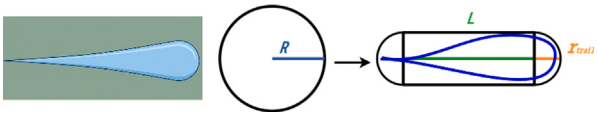


Fig. 5. Based on cartoon representations of water (left), expressive droplets are composed of a circular (resp. spherical in 3D) head and a tail of decreasing radius, whose center-line expresses the droplet's trajectory in the last time steps. The cartoon reference was generated with Nano Banana [1], to mimic the style of Tom & Jerry, the original artwork is not showed to respect the copyright.

4. Stylized effects

While their changes of state were inspired by prior work, visual particles now need to be used in novel ways to achieve the expressive rendering effects we aim for. Among the possible artistic representations, we decided to focus on the four typical effects illustrated in Fig. 2, chosen for their expressivity in animation and their aesthetics. 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

Our first goal is to create a representation for expressive droplets, inspired by the examples in Figs. 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 droplet's recent trajectory. The length of the tail seems to depend on the droplet's speed, as shown by some nearly circular droplets on top of Fig. 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_D state by creating a water density field – a scalar function decreasing with the distance – around a skeletal polyline generated from the trajectory of the particle in the last few time-steps (see Fig. 5, right). The skeleton joints J_i are computed as follows:

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

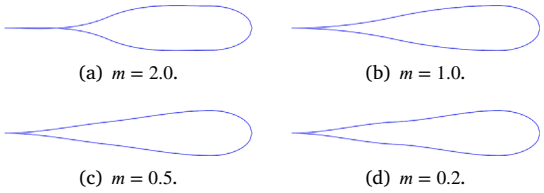


Fig. 6. Different styles of trails obtained by changing the middle point tangent m of the spline used for skeletal radius variation.

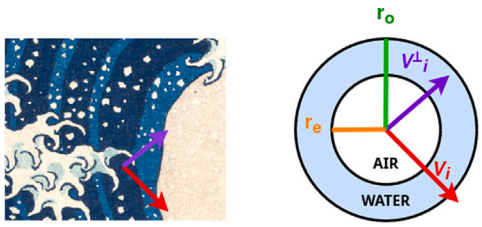


Fig. 7. To produce concave effects at wave crests (left), visual particles in S_C state distribute their air at the center (below r_e) and their water at the boundary (between r_e and r_o) of their volume (right). Their velocity v_i and perpendicular velocity $v_i^{\perp}$ are used to compute air entrainment.

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 Fig. 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 by 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_t is the radius attached to the head joint, 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 $0 < r_t \leq R$.

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

Once r_t 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_t 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, so that $\phi(0) = \phi_{\max}$ on the skeleton and $\phi(1) = 0$ at distance r_s. 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.

Fig. 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 most closely resemble those in Fig. 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_t in Eq. (3).

A few results are shown in Figs. 5 and 21, and 12 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 large water bodies) resemble a circle in 2D or a sphere in 3D, and keep the radius R.

4.2. Wave effects

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_C state: the trapped air is at the center, and is surrounded by a thin liquid layer (see Fig. 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:

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

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:

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

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_C 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_D state.

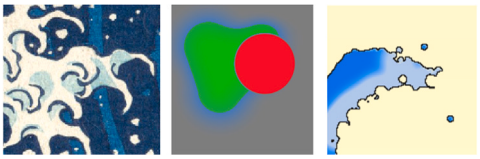


Fig. 8. 2D results of the concave effect. From left to right: reference image; a visual particle in concave state (red, usually not depicted) carving a body of liquid formed by particles in droplet state (green); the concave effect applied at a larger scale. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

We add another mechanism to make the effect still closer to the reference art-piece: Noting that the curve's 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:

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

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.

View-dependent effect. While the above method already generates the effects we seek in the 2D case, as shown in Fig. 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 carved perpendicular to the viewpoint direction used for rendering the image. To achieve this, we use this direction to project the effect into the camera plane as shown in Fig. 9. This is done as follows:

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

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 Eq. (7) replacing $\vec{v}_i$ and $\vec{v}_i^{\perp}$.

4.3. Underwater bubbles

A particle in S_B state shares a similar air–water distribution with those in the S_C 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 Fig. 2.

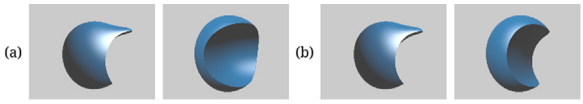


Fig. 9. Interaction between a 3D water droplet and a particle in concave state (not displayed but carving the first one) without (a) vs. with (b) a projected velocity over the camera plane. In the first sub-figures, the camera faces the plane where the two particles are located, requiring no correction.

We can also modify r_e and r_o computations to produce thinner or wider boundaries for the bubbles, using:

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

where γ is the bubble width modifier controlled by the user; r_e^* and r_o^* are then used in (7).

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 Fig. 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 (e.g., elongated shapes in the direction of the bubble 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 by a streak. We use white spheres 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 allows 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 as in Fig. 2. Taking these as inspiration, we aim to represent 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_F state are not necessarily 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 to the positions of the visual counterparts of the simulated particles, to project the foam above the surface.

To compute the bulky shapes, we set 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_B state may add air to the ones in S_F state. Therefore, the size of the foam bumps may increase over time.

5. 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. Finally, we also propose a technique to produce textures that enrich our stylized shapes, as described 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:

$$\begin{aligned}
\phi_{S_D} &= \sum \phi_{p \in S_D} \\
\phi_{S_C} &= \overset{*}{\max}\left(\phi_{p \in S_C}\right) \\
\phi_{S_B} &= \mathcal{O}\left(\phi_{p \in S_B}\right) \\
\phi_{S_F} &= \max\left(\phi_{p \in S_F}\right)
\end{aligned} \tag{13}$$

Note that the summation of fields, resulting in 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 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. For the bubbles, we keep the positive contributions where the gradients of the density function are negative or zero for the two objects. This is defined as follows:

$$\begin{aligned}
\overset{*}{\max}\left(\phi_{p_1}, \phi_{p_2}\right) &= \left\{ \begin{array}{cc} \phi_{p_1} & \text{if } |\phi_{p_1}| > |\phi_{p_2}| \\ \phi_{p_2} & \text{else} \end{array} \right\} \\
\mathcal{O}\left(\phi_{p_1}, \phi_{p_2}\right) &= \left\{ \begin{array}{cc} \phi_{p_1} + \phi_{p_2} & \text{if } \Delta\phi_{p_1,p_2} \leq 0 \\ \max(\phi_{p_1}, \phi_{p_2}) & \text{else} \end{array} \right\}
\end{aligned} \tag{14}$$

Alternatively, a simple $\max()$ operator can be used for the bubbles, depending on the desired style. An example of the difference in the result of the operators is shown in Fig. 20, where the choice of blending operator can produce smoother shapes for bubble clusters. The implicit shapes defined by combining these fields are then rendered as described next.

5.2. Stylized rendering

2D case. We use surface splatting to render different implicit shapes: One for water bodies, defined by the 0.5 iso-surface of $\phi_{S_D} + \phi_{S_C}$, one for bubbles defined as the 0.5 iso-surface of ϕ_{S_B}, and one for foam, defined as the 0.5 iso-surface of ϕ_{S_F}. 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 bubble streak, and optionally use the same mechanism to add a streak around concave and foam particles, if desired. Depending on the user's choice, streak regions can be rendered either under or on top of the other layers, as semi-transparent surfaces with no contour line.

Table 1
Variables used in stylized rendering for 3D scenes. Values from experiments as references for wave scenes.

#	Description	Value
h_{xyz}	Hit point on surface	–
$V_{air,h}$	Air volume around h_{xyz}	–
$\vec{n}$	Normal of surface	–
$\vec{w}$	Reference Axis	(0,1,0)
c_ϵ	Curvature Threshold	0.85
C_w	Crest of wave detector	–
W_w	Wall of wave detector	–
N_s	Near state to h_{xyz}	–
h_s^c	Point at crest	–
h_s^w	Point at wall	–
h_s^s	Point elsewhere	–

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 $\phi_{S_D} + \phi_{S_C}$, and on the other hand the foam surface defined by the 0.5 iso-surface of ϕ_{S_F}. 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 ϕ_{S_B}. In our implementation, we used ray-casting for rendering.

5.3. Stylized surface pattern

Beyond geometry, the visual appearance of 3D stylized surfaces can be further enriched by adapting the surface material. Building on the particle state information from our multi-material model, we propose a spatially varying appearance based on the Blinn–Phong diffuse model. We illustrate this approach on breaking-wave scenes, where stylized artworks exhibit a characteristic crest/wall color separation: we distinguish the crest (the top of the wave) from the wall (its near-vertical front face). The formulation we describe below can serve as a template for other scene types, but the parameters and orientation conventions reported here are specifically tuned for breaking waves, and the pattern can be fully disabled when a simpler rendering is preferred. We introduce a two-step modification of the standard diffuse computation that produces visual patterns characteristic of stylized artworks, as formulated in (15) with variables defined in Table 1.

$$\begin{aligned}
C_w &= V_{air,h}^8 > 0 \text{ and } n \cdot w \geq c_\epsilon \\
W_w &= V_{air,h} > 0 \text{ and } n \cdot w < c_\epsilon \\
h_s^* &= \left\{ \begin{array}{cc} h_s^c \text{ if} & N_s = S_C \text{ or } C_w \\ h_s^w \text{ if} & W_w \\ h_s^s \text{ else} & \end{array} \right\}
\end{aligned} \tag{15}$$

This equation computes the scene segmentation h_s^*, classifying each surface hit point h_{xyz} based on two criteria: the local air volume $V_{air,h}$ and the orientation of the surface normal $\vec{n}$ relative to the reference axis $\vec{w}$. We estimate $V_{air,h}$ at the hit point by aggregating, with a compact kernel evaluated at h_{xyz}, the air content of nearby visual particles in the S_C and S_B states. The exponent on $V_{air,h}$ in the crest condition emphasizes regions of high local air content; in practice both conditions are evaluated against a small numerical tolerance rather than strict zero. The threshold $c_\epsilon = 0.85$ corresponds to an angular tolerance of about 32° between the surface normal and the reference axis: surfaces within this tolerance are classified as crest, others as wall.

The nearest particle state N_s is identified using an anisotropic distance with weights $(\frac{1}{32}, \frac{1}{4}, \frac{1}{32})$ on (x, y, z) differences. These weights reflect the typical geometry of breaking waves, where the vertical direction carries the dominant structural information; their values can be adapted to other scene types or orientations. The vertical weight is applied only when the y-difference is positive, giving priority to particles located above the hit point, where concave effects are most prominent.

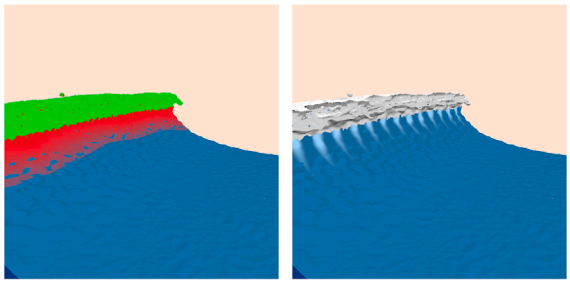


Fig. 10. Left: segmentation of the wave surface into crest regions (green, h_s^c) and wall regions (red, h_s^w). Right: resulting stylized rendering obtained by applying the color pattern according to this segmentation. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The coloring is then determined by the assigned region. For crest points h_s^c, a spatially decaying blend is applied: the air color is used at h_{xyz} and smoothly transitions to the water color with increasing distance; a hard cutoff can optionally be introduced to reproduce the sharp color separations characteristic of stylized wave artworks. For wall points h_s^w, we apply a spatially oscillating pattern I_{air}, defined as:

$$I_{air} = V_{air,h} \cdot \left(\frac{\cos(8\pi h_z)}{2} + \frac{1}{2}\right) \tag{16}$$

where I_{air} is modulated by the local air volume $V_{air,h}$ and varies periodically along the z-axis. The factor 8π controls the spatial frequency of the bands (4 oscillations per unit length in scene units), chosen to match the band density observed in the reference artworks; this value, like the other parameters of this section, can be adjusted to fit a different reference. The pattern reproduces the layered color patterns observed in stylized artworks such as Fig. 2(a). Two user-defined colors are linearly interpolated using I_{air}. For surface points h_s^s, the water color is used by default; the same pattern can optionally generate a white-caps effect. The resulting segmentation and final rendering are shown in Fig. 10.

This pattern is flexible: its thresholds and anisotropic distance weights can be adjusted to produce a variety of surface appearances, and the pattern layer can be fully disabled when a simpler rendering is preferred.

6. Results and discussions

6.1. Implementation and performance

We used the tool SPlisHSPlasH [36] 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 an NVIDIA RTX 4070 GPU, with uniform grids used as acceleration structures mainly in 3D scenes. Table 2 reports the rendering time per frame for different test scenes, with and without a uniform grid acceleration structure.

For 2D scenes, interactive frame rates are achieved even without an acceleration structure. For 3D scenes, an acceleration structure is required to reach interactive rates; however, such structures can be sensitive to highly irregular particle distributions or large primitives like long skeletons for the droplets, which may affect performance. Changing the acceleration structure to a bounding volume hierarchy (BVH) could also help with these issues, but this is out of scope for this paper. Rendering times also vary with camera position, as the amount of processed data depends on the visible scene extent.

Table 2
Rendering performance for different test scenes. For each scene, we report the number of simulated particles, whether a uniform grid acceleration structure was used (Accel.), and the average time per frame (TPF) in seconds.

Test	# Particles	Accel.	TPF [s]
Bubbles 2D	~10K	No	0.66
Fountain 2D	~20K	No	0.126
Fountain 3D	~1K	No	11.16
Wave 3D	~82K	Yes	1.72
Wave 3D	~160K	Yes	2.76
Wave 3D	~190K	Yes	3.53

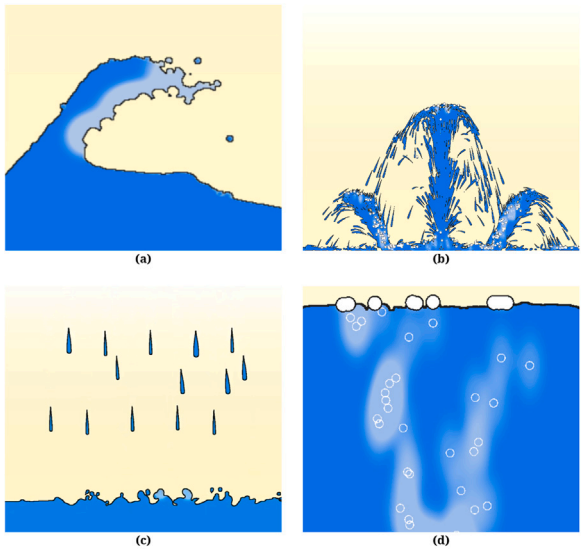


Fig. 11. Collection of 2D visualizations obtained with our method, here we show: (a) Crest of wave, (b) Fountain, (c) Rain droplets and (d) Column of bubbles going to surface. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

6.2. 2D results

Fig. 11 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 Fig. 11(a)), which only disappear when the wave collapses and concave particles switch to the bubble state (see Fig. 22). In this first example, only S_D and S_C were activated to make the result similar to the Japanese painting reference, but we used a streak generated by concave particles to add a semi-transparent white coloring over the crest of the wave. In the fountain example (see Figs. 11(b) and 21), the use of the expressive droplets achieves cartoon-like rendering, and parameters were set to generate curved tails that resemble the ones observed in Fig. 2. The same model was also applied to a rainy scene, illustrated in Figs. 11(c) and 19, and to a 2D splash scene (Fig. 18), where the droplet tails curve to follow the particle trajectories. Finally Figs. 11(d) and 20 show a column of rising bubbles transforming into foam, inspired by Fig. 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, Fig. 17 shows different renderings of the same bubbles scene obtained by editing the shape of the streak.

6.3. 3D results

Some of our renderings of 3D animations are gathered in Fig. 12: (a) shows the same wave crest as in Fig. 1, but from a different viewing

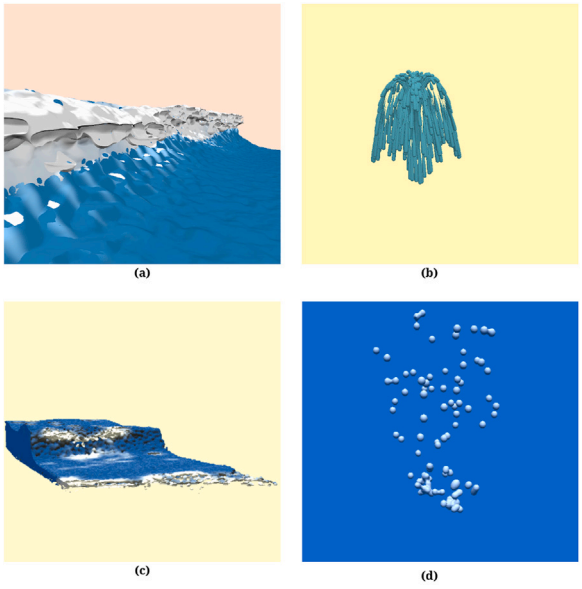


Fig. 12. Display of different 3D visualization obtained with our method. From left to right we have: (a) Crest of a big wave, (b) Droplets falling, (c) Offshore small wave, (d) Column of bubbles.

angle, showing that the 3D concave effect remains salient whatever the camera viewpoint, thanks to the adaptation. Fig. 24 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 Figs. 12(d) and 23 an example of an underwater scenario where a column of air is modeled thanks to the advection of the air component of visual particles in bubble state.

6.4. Perceptual evaluation

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 [37] and DomoAI [38]. 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.

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 [37]. 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 [38] and Firefly, removing any detailed motion description from the prompt but providing the video of particle 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.

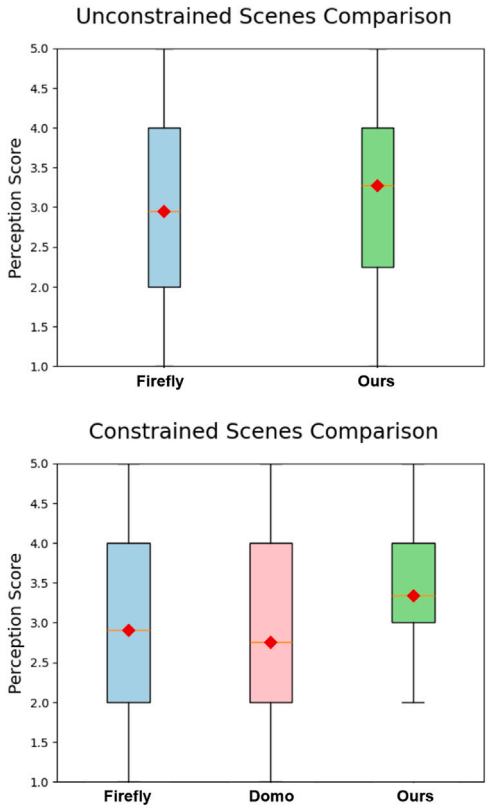


Fig. 13. User study results for Unconstrained scenes (top) versus constrained scenes where the AI was given our input particle-based animation clip for extra conditioning (bottom). Higher score means better, the mean being represented by the red diamond. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 3
User study perception score per question, with from top to bottom Unconstrained (Q1-6) and then Constrained (Q7-11) comparisons. The best (green) scores are highlighted in each segment for every method.

#	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

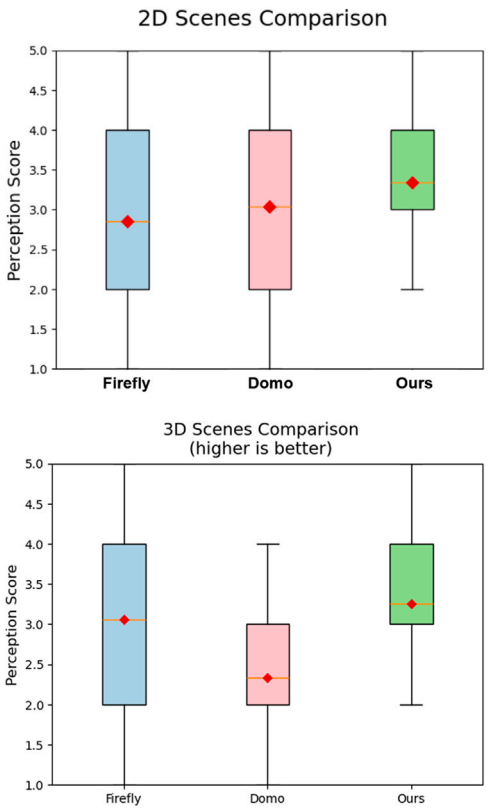


Fig. 14. The same results are analyzed according to the 2D or 3D nature of the scene. Higher means better, the mean being represented by the red diamond.

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 US and CS cases. A summary of the survey results can be observed in Table 3, Figs. 13 and 14. 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 the US our method performs better than Firefly, while in the CS our method also performs better than both Firefly and Domo. Taking into account the Wilcoxon test results, we confirm that our method indeed better captures the style of the reference images in the chosen scenarios. The results in Fig. 14 show 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 perception score of 3.34 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, such as 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

Fig. 15. Camera movement around a breaking wave: (a) Firefly (Constrained), (b) our method. Our approach preserves geometric consistency regardless of viewpoint, while the AI-generated geometry breaks down during rotation, as shown by the disappearance of the water bed.

thanks to style transfer and to the visual elements they added based on the prompt, such as 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 famous Japanese 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. Geometric consistency under camera movement

Beyond the perceptual scores reported above, we consistently observed a difficulty with Firefly: maintaining geometric consistency under camera movement. As illustrated in Fig. 15, rotating the camera around the scene causes the geometry produced by Firefly to break down, with parts of the wave surface clearly missing, and even small adjustments to the initial camera position can produce entirely different results. We consistently observed this behavior across our tests with Firefly; we have not benchmarked all current generative video tools, but similar inconsistencies are likely to occur with tools that do not rely on an explicit 3D representation. In contrast, our method, being grounded in an explicit particle-based simulation, naturally produces geometrically consistent outputs across viewpoints and over time.

6.6. qualitative case study with a professional artist

We conducted a qualitative case study with a professional 2D artist to assess our method as a practical animation tool and gather practitioner feedback. The artist was asked to produce three stylized animations – a crushing wave, a column of bubbles, and a fountain of water – first using our tool, then recreating the same scenes with their standard workflow. For each scene, the same reference artwork, target sequence duration, and stylistic intent were used in both conditions, and the artist worked until they considered the result satisfactory with respect to the reference; they were free to stop earlier if needed and

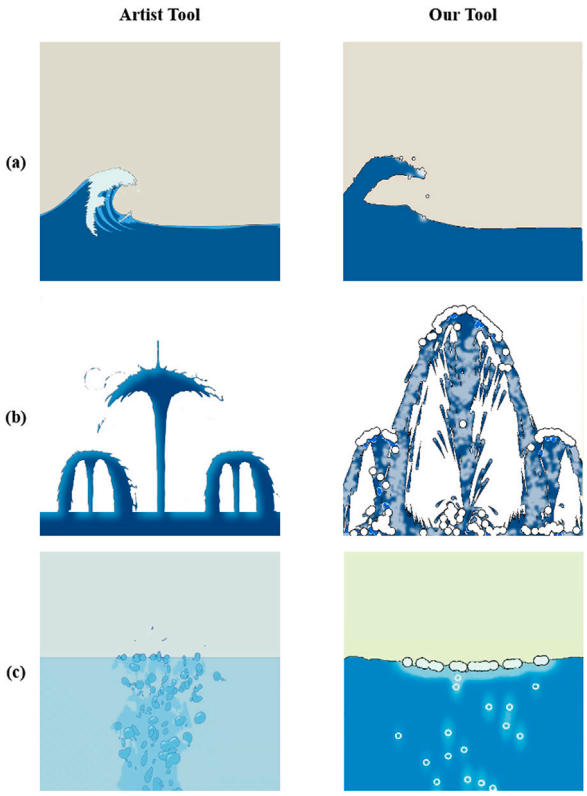


Fig. 16. Side-by-side comparison of results produced by a professional 2D artist using Toon Boom Harmony (left) and our tool (right) for three scenes: (a) Crushing Wave, (b) Fountain, and (c) Bubbles.

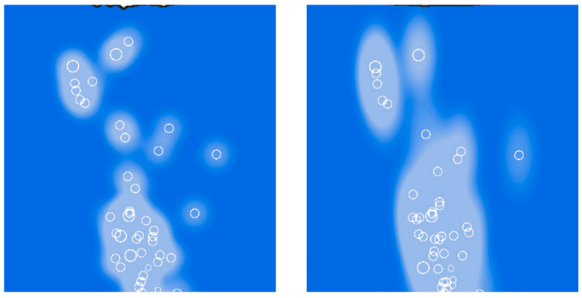


Fig. 17. Tuning the bubble streak. Comparison between a circular (left) and an ellipsoidal kernel oriented in the velocity direction of each particle in S_B state (right).

Fig. 18. 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.

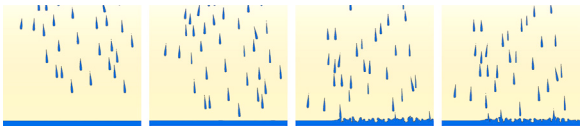


Fig. 19. 2D animation of rain splashing on a water surface.

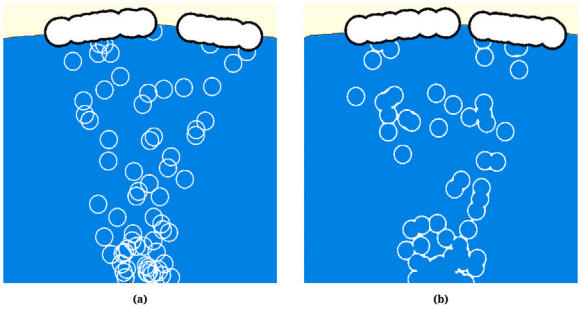


Fig. 20. A 2D under-water air column. Bubbles propagate upward even if particles do not move, thanks to the propagation of bubble state. Foam appears when bubbles reach the surface. We show the results using the (a) $max()$ and the (b) $\mathcal{O}()$ operators , please note the difference in the clustered regions, from the smoother blending in (b) compared to (a).

Table 4
Time required by the professional artist to complete each scene with Toon Boom Harmony and our tool, in hours (h) and minutes (m).

Tool	Scene	Time
Toon Boom	Fountain 2D	4 h 53 m
Toon Boom	Bubbles 2D	4 h 35 m
Toon Boom	Wave 2D	48 h 35 m
Ours	Fountain 2D	44 m
Ours	Bubbles 2D	35 m
Ours	Wave 2D	47 m

they were also allowed to look for more reference images. For the standard workflow, the artist chose Toon Boom Harmony [39], using onion skinning to maintain temporal consistency across frames. The results are shown side by side in Fig. 16.

Before the experiment, the artist participated in a 15-minute guided training session where an explanation of the interface was given, including the intended visual effect of each editable parameter, and a live demonstration of how to use the tool with a sample scene. After the training session, the artist completed all three scenes in about 45 min each, producing multiple stylized alternatives by tweaking the editable parameters within that time (Table 4). In contrast, on this small sample and despite extensive experience with Toon Boom Harmony, the same artist required between 4.5 h and 2 days to reach a result they considered satisfactory with their standard workflow — an order of magnitude longer than with our tool.

A discussion session was held following the experiment. The artist particularly emphasized the value of our tool for modeling expressive water fountains with elongated droplets — a scenario they found especially difficult to achieve with traditional methods, and which our approach handles naturally in both 2D and 3D. While our tool produced first viable results substantially faster across all scenes in this case study, the artist also noted a reduced sense of creative control due to the degree of automation involved and expressed the fact that additional parameters would be necessary to further fine-tune the expressive effects. Some suggestions are straightforward to incorporate , such as being able to define the base shape for the bubbles; others, such as personalized flow-guided animation control, are harder to address given our reliance on pre-computed simulations. The artist also suggested adding a built-in preview feature. Currently, our pipeline relies on FFmpeg via the command line to generate video outputs. Because of this, we had to manually intervene during the experiment to render these previews, as the command-line process fell outside the artist's standard workflow.

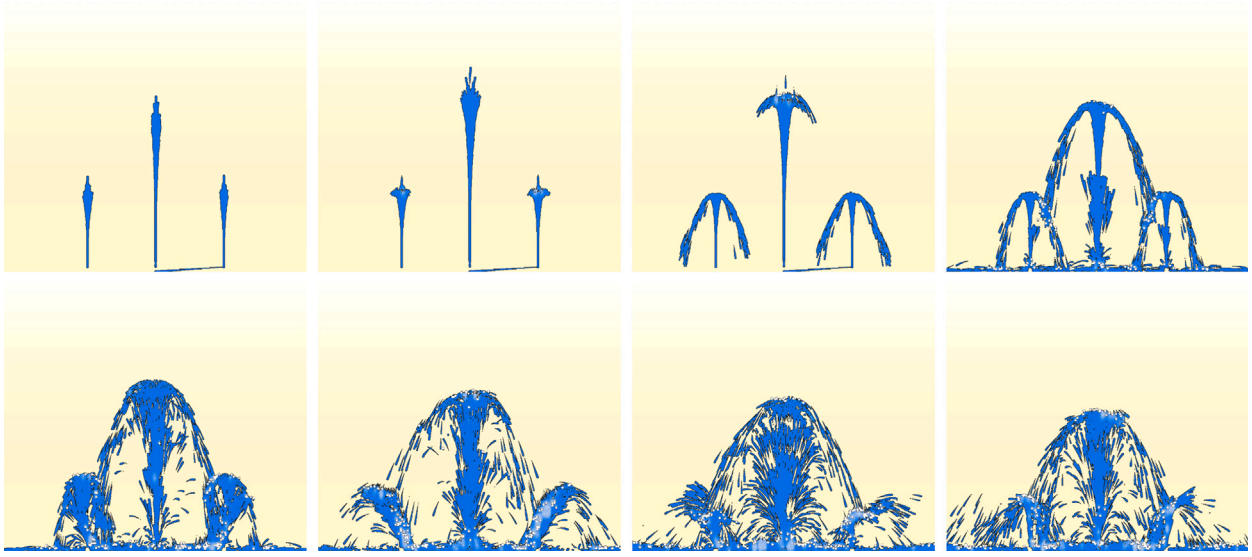


Fig. 21. Sequence of a fountain 2D animation (left to right, top to bottom) showing the mixture of effects from our method over an interval between the start of the fluid emission and the mixture of columns of fluids. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

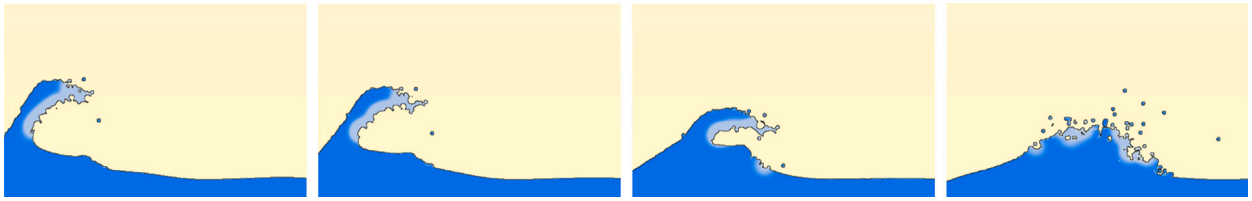


Fig. 22. 2D animation of the collapse of a wave, simulated using 26 K particles. Particles in concave state at the crest of the wave are used both to dig cavities and to generate a white streak opposite to their velocity direction, to emphasize turbulent motion at the crest. Here, the elongated droplets effect was disabled, since circular droplets better resemble the artwork that inspired us (shown in Fig. 1, left).

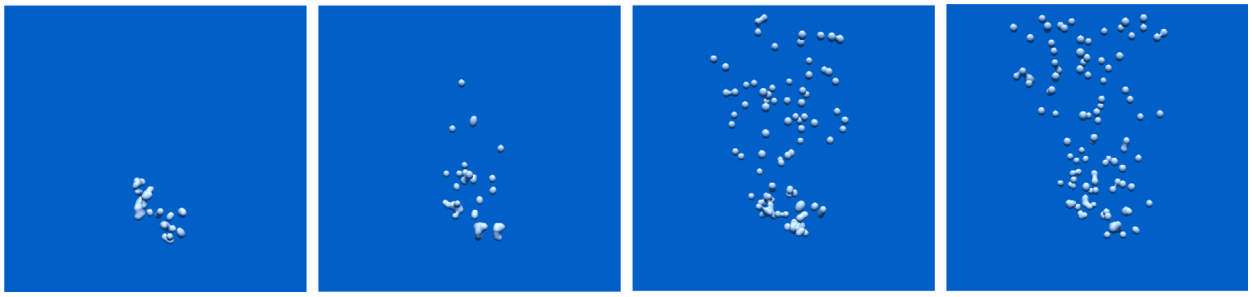


Fig. 23. A 3D underwater scene with a column of bubbles modeled with our method.

6.7. 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 Eq. (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 neighbor.

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.

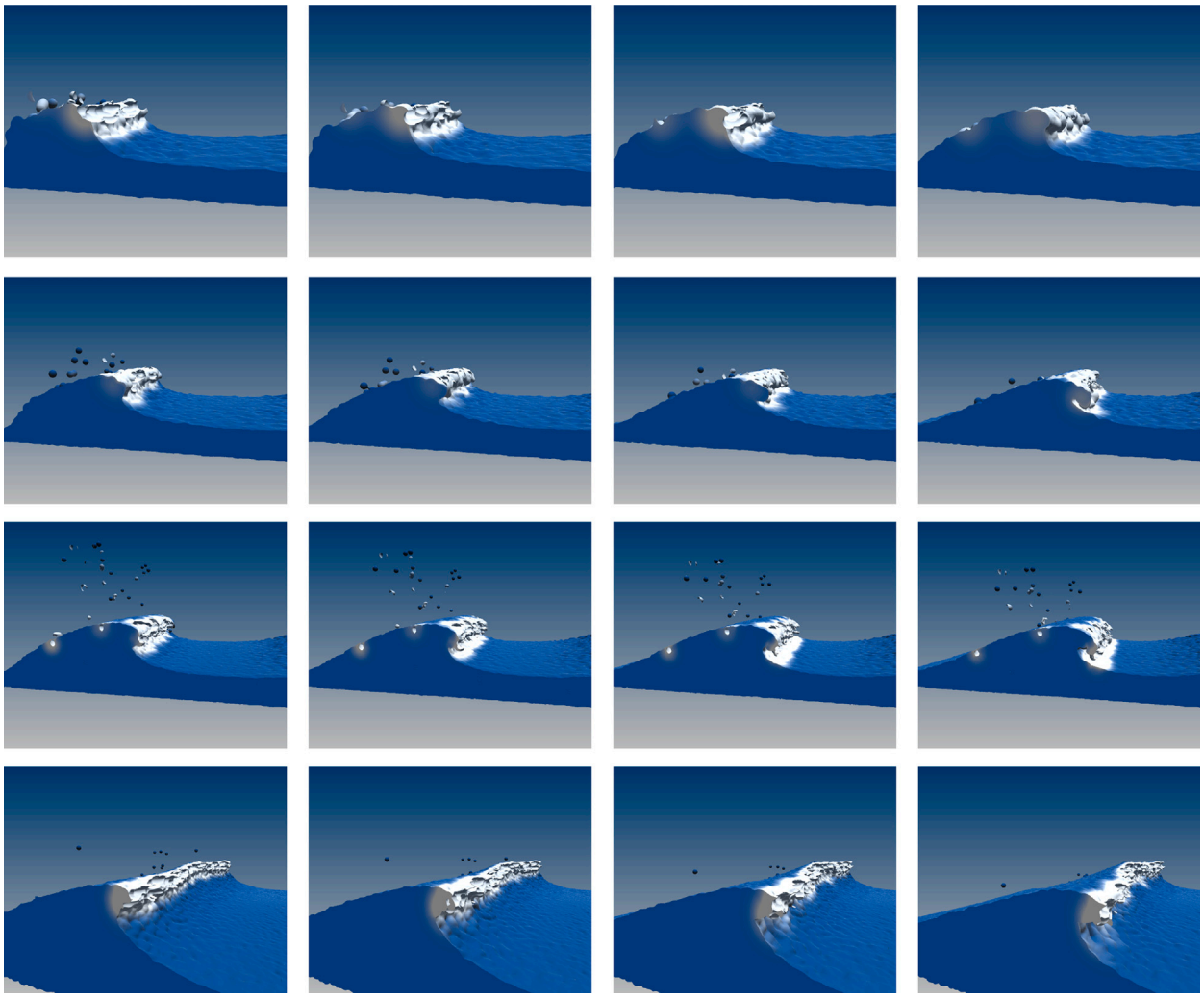


Fig. 24. Another wave sequence, illustrating the moment between the climax of the wave and the beginning of the collapse. As shown from top to bottom, the spatial resolution of the simulation plays a strong role in the visual results: We respectively used ∼10K particles (top), ∼24K particles (second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

Lastly, while our method can achieve more convincing results than the AI tools, the latter can still switch easily to different artistic styles so they are more widely applicable than our current method. While the identified effects might be enough to produce stylized animations, our procedural approach requires considering the different styles that these effects could have, and to develop dedicated methods.

Finally, our case study with a professional artist (Section 6.6) involves a single practitioner; a larger study with multiple artists and a third-party quality rating would be required to generalize these observations.

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 artworks. Moreover, it is versatile and can be easily applied to both 2D and 3D input animations. Our method builds on any existing particle-based simulation and applies a dedicated post-processing layer to generate stylized effects, with no added complexity to the underlying simulation. We further introduced a stylized surface-color pattern demonstrated on breaking-wave scenes, validated our perceptual results against generative-AI tools, and reported a hands-on case study with a professional 2D artist.

In future work, we plan to improve the rendering of some of the effects, such as the behavior of bubbles reaching the surface level, by extending recent models [40,41] to bubbles and foam in a cartoon style. Moreover, while we limited our state machine to a set of four pre-defined stylization effects (namely droplets, concave shapes, bubbles, 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, drawing inspiration from recent sketch-based solutions in implicit modeling [28]. This can further extend the range of artistic styles that our method could handle. The stylized surface-color pattern could similarly be generalized beyond breaking-wave scenes, and the case study with a professional artist suggests that a larger-scale evaluation involving multiple practitioners would be a valuable next step.

CRediT authorship contribution statement

Rodrigo Stevenson-Regla: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation. **Damien Rohmer:** Writing – review & editing, Writing – original draft, Supervision, Methodology, Funding acquisition. **Loïc**

Barthe: Writing – review & editing, Supervision, Funding acquisition. **Marie-Paule Cani:** Writing – review & editing, Writing – original draft, Supervision, Methodology.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We thank the reviewers for their constructive feedback, which substantially improved the quality of this paper. We also extend our gratitude to **Nicolas Escande** for providing valuable insights and feedback from a professional artist's perspective. This project was funded by the ANR-22-CE33-0018 Multiform grant.

Data availability

Data will be made available on request.

References

[1] Google, Nano banana 2: Gemini AI image generator & photo editor, 2026, https://gemini.google/overview/image-generation/. AI Generation Tool.
[2] R. Stevenson-Regla, D. Rohmer, L. Barthe, M.-P. Cani, Implicit field-based stylization of 2D and 3D liquid animations, in: M. Comino Trinidad, C. Mancinelli, F. Maggioli, C. Romanengo, D. Cabiddu, D. Giorgi (Eds.), Smart Tools and Applications in Graphics - Eurographics Italian Chapter Conference, The Eurographics Association, 2025, http://dx.doi.org/10.2312/stag.20251324.
[3] M. Droske, J. Hanika, J. Vorba, A. Weidlich, M. Sabbadin, Path tracing in production: The path of water, in: ACM SIGGRAPH 2023 Courses, SIGGRAPH '23, ACM, New York, NY, USA, 2023.
[4] N. Ang, A. Catling, F.C. Ciardi, V. Kozin, The technical art of sea of thieves, in: ACM SIGGRAPH 2018 Talks, SIGGRAPH '18, ACM, New York, NY, USA, 2018.
[5] R. Bridson, Fluid Simulation for Computer Graphics, second ed., A K Peters/CRC Press, 2015.
[6] M. Ihmsen, N. Akinci, G. Akinci, M. Teschner, Unified spray, foam and air bubbles for particle-based fluids, Vis. Comput. 28 (6–8) (2012) 669–677.
[7] J. Bender, D. Koschier, T. Kugelstadt, M. Weiler, Turbulent micropolar SPH fluids with foam, IEEE Trans. Vis. Comput. Graphics 25 (6) (2019) 2284–2295.
[8] J. Wretborn, S. Flynn, A. Stomakhin, Guided bubbles and wet foam for realistic whitewater simulation, ACM Trans. Graph. 41 (4) (2022).
[9] S. Baek, K. Um, J. Han, Muddy water animation with different details, Comput. Animat. Virtual Worlds 26 (3–4) (2015) 347–355.
[10] M. Ihmsen, J. Bader, G. Akinci, M. Teschner, Animation of air bubbles with SPH, in: International Conference on Computer Graphics Theory and Applications, 2011.
[11] J. Wretborn, A. Stomakhin, C. Batty, A unified multi-scale method for simulating immersed bubbles, Comput. Graph. Forum (2025).
[12] A.M. Eden, A.W. Bargteil, T.G. Goktekin, S.B. Eisinger, J.F. O'Brien, A method for cartoon-style rendering of liquid animations, in: Proceedings of Graphics Interface 2007, ACM, New York, NY, USA, 2007.
[13] J. Liao, J. Yu, J. Patterson, Modeling ocean waves and interaction between objects and ocean water for cartoon animation, Comput. Animat. Virtual Worlds 22 (2–3) (2011) 81–89.
[14] S. Liu, W. Zhang, Cartoon-style simulation of water coupled with objects, Simul. Model. Pr. Theory 31 (2013).
[15] L.d.S. Rocha Neto, A.L. Apolinario, Cartoon water rendering with foam and surface smoothing, in: 2014 Brazilian Symposium on Computer Games and Digital Entertainment, 2014.
[16] M. Browning, C. Barnes, S. Ritter, A. Finkelstein, Stylized keyframe animation of fluid simulations, in: Proceedings of the Workshop on Non-Photorealistic Animation and Rendering, NPAR '14, ACM, 2014.
[17] J. Lee, J.-H. Kim, H.-Y. Lee, S.-J. Kim, Realistic fluid representation by anisotropic particle, J. Vis. 22 (2) (2019).
[18] B. Kim, V.C. Azevedo, M. Gross, B. Solenthaler, Lagrangian neural style transfer for fluids, ACM Trans. Graph. 39 (4) (2020).
[19] P. Kanyuk, V. Azevedo, R. Ortiz, J. Tang, Singed silhouettes and feed forward flames: Volumetric neural style transfer for expressive fire simulation, in: ACM SIGGRAPH 2023 Talks, SIGGRAPH '23, Association for Computing Machinery, New York, NY, USA, 2023, http://dx.doi.org/10.1145/3587421.3595435.
[20] J. Tang, B. Kim, V.C. Azevedo, B. Solenthaler, Physics-informed neural corrector for deformation-based fluid control, Comput. Graph. Forum 42 (2) (2023) 161–173, http://dx.doi.org/10.1111/cgf.14751, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14751. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14751.
[21] Y. Deng, H.-X. Yu, D. Zhang, J. Wu, B. Zhu, Fluid simulation on neural flow maps, ACM Trans. Graph. 42 (6) (2023) http://dx.doi.org/10.1145/3618392.
[22] Y. Chen, G. Shao, K.C. Shum, B.-S. Hua, S.-K. Yeung, Advances in 3D neural stylization: A survey, Int. J. Comput. Vis. 133 (8) (2025) 5026–5061, http://dx.doi.org/10.1007/s11263-025-02403-9.
[23] J. Bloomenthal, C. Bajaj, J. Blinn, M.-P. Cani, A. Rockwood, B. Wyvill, G. Wyvill, Introduction to Implicit Surfaces, Morgan Kaufmann, 1997.
[24] M.-P. Cani, M. Desbrun, Animation of deformable models using implicit surfaces, IEEE Trans. Vis. Comput. Graphics 3 (1) (1997) 39–50, Published under the name Marie-Paule Cani-Gascuel.
[25] A. Opalach, M.-P. Cani, Local deformation for animation of implicit surfaces, in: W. Straßer (Ed.), Spring Conference on Computer Graphics, SCCG, Bratislava, Slovakia, 1997, Published under the name Marie-Paule Cani-Gascuel.
[26] F. Canezin, G. Guennebaud, L. Barthe, Topology-aware neighborhoods for point-based simulation and reconstruction, in: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '16, 2016, pp. 37–47.
[27] F. Cordier, K. Singh, Y. Gingold, M.-P. Cani, Sketch-based modeling, in: Siggraph Asia 2016, in: SIGGRAPH ASIA 2016 Courses, ACM, Macao, China, 2016, p. 222.
[28] B. Angles, M. Tarini, B. Wyvill, L. Barthe, A. Tagliasacchi, Sketch-based implicit blending, ACM Trans. Graph. 36 (6) (2017).
[29] O. Gourmel, L. Barthe, M.-P. Cani, B. Wyvill, A. Bernhardt, M. Paulin, H. Grasberger, A gradient-based implicit blend, ACM Trans. Graph. 32 (2) (2013).
[30] R.H. Kazi, F. Chevalier, T. Grossman, S. Zhao, G. Fitzmaurice, Draco: Bringing life to illustrations with kinetic textures, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '14, Association for Computing Machinery, New York, NY, USA, 2014, pp. 351–360, http://dx.doi.org/10.1145/2556288.2556987.
[31] O. Jamriška, J. Fišer, P. Asente, J. Lu, E. Shechtman, D. Sýkora, LazyFluids: Appearance transfer for fluid animations, ACM Trans. Graph. 34 (4) (2015) http://dx.doi.org/10.1145/2766983.
[32] J. Xing, R.H. Kazi, T. Grossman, L.-Y. Wei, J. Stam, G. Fitzmaurice, Energybrushes: Interactive tools for illustrating stylized elemental dynamics, in: Proceedings of the 29th Annual Symposium on User Interface Software and Technology, UIST '16, Association for Computing Machinery, New York, NY, USA, 2016, pp. 755–766, http://dx.doi.org/10.1145/2984511.2984585.
[33] J. Ma, L.-Y. Wei, R.H. Kazi, A layered authoring tool for stylized 3D animations, in: Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems, CHI '22, Association for Computing Machinery, New York, NY, USA, 2022, http://dx.doi.org/10.1145/3491102.3501894.
[34] P. Olivier, T. Butler, P. Guehl, J.-L. Coll, R. Chabrier, P. Memari, M.-P. Cani, DynBioSketch: A tool for sketching dynamic visual summaries in biology, and its application to infection phenomena, Comput. Graph. 122 (2024) 103956, http://dx.doi.org/10.1016/j.cag.2024.103956, URL: https://hal.science/hal-04682753.
[35] H. Xie, K. Arihara, S. Sato, K. Miyata, DualSmoke: Sketch-based smoke illustration design with two-stage generative model, Comput. Vis. Media 10 (5) (2024) 965–979, http://dx.doi.org/10.1007/s41095-022-0318-0.
[36] J. Bender, et al., SPlisHSPlasH library, 2016.
[37] A. Adobe, Adobe Firefly, Image Model 4 [Generative AI model], Adobe Systems Incorporated, 2025, https://www.adobe.com/products/firefly.html. AI Generation Tool.
[38] DomoAI, DomoAI: Video-to-video, style transfer, AI, video tool, 2025, https://www.domoai.app/. AI Generation Tool.
[39] Toon Boom Animation Inc., Toon boom harmony (version 25), 2025, URL: https://www.toonboom.com/products/harmony. Professional 2D Animation Software.
[40] O. Atasi, D. Legendre, B. Haut, R. Zenit, B. Scheid, Lifetime of surface bubbles in surfactant solutions, Langmuir 36 (27) (2020) 7749–7764.
[41] J. Miguet, F. Rouyer, E. Rio, The life of a surface bubble, Molecules 26 (5) (2021).