The Visual Computer
https://doi.org/10.1007/s00371-021-02208-0

ORIGINAL ARTICLE

Real-time simulation of violent boiling in concentrated sulfuric acid dilution

Wentao Chen[1] · Tian Sang[1] · Yitian Ma[1] · Qian Chen[1] · Yuwei Xiao[1] · Zhigeng Pan[2] · Xubo Yang[1]

Accepted: 4 June 2021

Abstract
Concentrated sulfuric acid dilution is an essential yet hazardous experiment in chemistry education. Incorrect operations can lead to intensive boiling and splattering or even endanger the experiment operator. This paper presents a novel approach to simulate the violent boiling phenomenon in real time and an efficient method to deal with particle penetration caused by fast motion. We introduce the anti-viscosity method to inject momentum into high-temperature particles and capture the chaotic dynamics in the boiling process with small computational overhead. To address the particle penetration issue, we propose a new constraint type called flask constraint which uses radius lookup tables to represent axisymmetric shapes efficiently and projects the particles back into their container when penetration occurs. These methods are integrated into a virtual reality application to simulate the concentrated sulfuric acid dilution experiment and demonstrate the efficiency and effectiveness of our method. Our work improves the capability and stability of particle-based fluid solvers and provides appealing solutions for integrating fluid simulation into interactive applications.

Keywords Violent boiling simulation · Real-time simulation · Fluid simulation · Position-based fluids · Virtual experiment

1 Introduction

Diluting concentrated sulfuric acid is a common experiment in chemistry education. However, few students have carried out this experiment since working with concentrated acids is quite dangerous. Sulfuric acid is extremely corrosive and capable of causing severe burns. Diluting acid can get even more dangerous due to the large quantity of heat generated in the dilution process. During sulfuric acid dilution, it is essential that the sulfuric acid is added to the water instead of the other way around. The addition of water to sulfuric acid can produce a large amount of heat to boil the diluted acid or even lead to an explosion when working with highly concentrated acid.

In the face of this experiment's hazardous nature, we develop real-time simulation techniques and build a virtual reality (VR) application for concentrated sulfuric acid dilution, enabling students to conduct experiments in a safe environment. Our system provides simulation results for both correct and incorrect operations. Students can observe the dangerous consequences of incorrect operations and learn from their failures, thus improving safety awareness and preparing for real experiments.

The most critical and challenging part of our system is to simulate the violent boiling process. Previous researches focus on modeling the liquid–gas interaction in order to simulate the boiling behavior. In 2005, Müller et al. [28] proposed a bubble-based technique to simulate boiling water by randomly generating air particles on the caviation sites. Later

✉ Xubo Yang
yangxubo@sjtu.edu.cn

Wentao Chen
vitalight@sjtu.edu.cn

Tian Sang
sangtian0820@sjtu.edu.cn

Yitian Ma
mkochab@sjtu.edu.cn

Qian Chen
kcsnow@sjtu.edu.cn

Yuwei Xiao
xiaoyuwei@sjtu.edu.cn

Zhigeng Pan
zgpan@hznu.edu.cn

[1] Digital ART Laboratory, Shanghai Jiao Tong University, Shanghai, China

[2] School of Artificial Intelligence, Nanjing University of Science and Technology, Nanjing, China

Published online: 10 July 2021

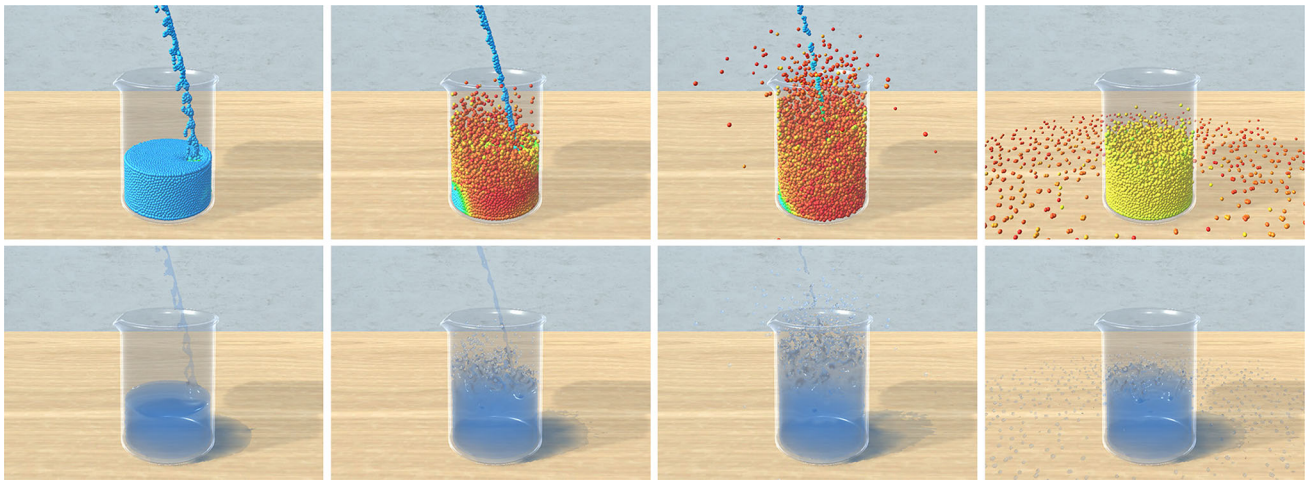


Fig. 1 Simulation result of pouring water into concentrated sulfuric acid over time. A few seconds after pouring, the acid heats up rapidly and presents increasingly violent boiling behavior. Eventually, the acid cools down and gradually stabilizes. The top row visualizes the fluid particles colored by their temperature. The bottom row shows the result simulated and rendered in real time

works focused on improving the simulation detail of the boiling process, such as scaling and bursting of bubbles [31], deformation and condensation [11], improving mass and heat transfer model [16]. These methods focus on simulating boiling phenomena with external heat sources and relatively calm fluid surfaces, in which case modeling bubble is sufficient to provide visual plausibility. However, it is difficult to simulate intensive boiling and violent splashing with these methods. Bubbles get removed when they reach the fluid surface and cannot bring enough momentum to the liquid to produce highly chaotic motion. Moreover, bubbles introduce extra modeling complexity and require lots of computational resources, which is undesirable for real-time applications.

In this work, we propose a bubble-free approach to simulate the violent boiling process. First, the acid concentration and heat are diffused between particles. Then, the heat produced by diluting concentrated sulfuric acid is computed and distributed to fluid particles. We further use a model named *anti-viscosity*, which injects momentum into particles with high temperatures, thus producing the violent boiling behavior. Our boiling simulation method does not involve bubbles and works directly on liquid particles. In this way, our method is easier to implement, is more efficient and has better control over the liquid motion. The simulation result of diluting concentrated sulfuric acid is demonstrated in Fig. 1. Our anti-viscosity model is general enough to model other kinds of boiling phenomena such as boiling water, as shown in Fig. 7.

Although fluid simulators reached real-time performance years ago, they have rarely been integrated into interactive applications. One of the reasons behind this is the poor stability when user input gets involved with fluid simulation. Though physics simulation techniques such as position-based dynamics (PBD) [27] work well in most cases, they may fail to remain stable in interactive applications where users can move objects with arbitrary velocity and acceleration. The constraints in PBD are not designed to cope with user interference, which can be especially problematic for fluid simulators in VR applications. For instance, if the user is holding a container full of liquid and moves it with a relatively large acceleration, discrete collision detection will be missed. As a result, a large number of fluid particles will penetrate through the container and spill on the ground, leading to an unpleasant user experience.

To address the particle penetration issue, we introduce *flask constraint* into our system. Analytical expressions are devised to represent the shape of flask containers and are used to detect and solve particle penetrations. Flask constraint works for any container with an axisymmetric shape and can remedy against particle penetrations with little performance overhead. Moreover, flask constraint can be used alongside traditional collision handling methods such that solid–fluid coupling can still be physically simulated when there is no extreme movement involved.

The main contributions of this paper can be summarized as:

- A real-time approach to capture the chaotic dynamics in the violent boiling process. Our approach is highly efficient and can model boiling phenomena with different levels of intensity.
- A new constraint type called *flask constraint* to remedy against particle penetration under extreme movement, which allows robust user interaction with simulated fluids.

– A virtual reality educational application that integrated our methods and simulates the process of diluting concentrated sulfuric acid. Our application provides inquiry-based learning by allowing users to observe different results produced by different operations.

2 Related work

2.1 Fluid simulation

The simulation of fluids has been a popular research topic in computer graphics for decades. Smoothed particle hydrodynamics (SPH) was first developed for the simulation of astrophysical problems [17] and then extended to model a wide range of problems including deformable bodies [7] and fluids [26]. Several extensions are proposed to improve performance and incompressibility, such as predictive–corrective incompressible SPH (PCISPH) [33], implicit incompressible SPH (IISPH) [12], position-based fluids (PBF) [19] and divergence-free SPH (DFSPH) [3]. Other works focus on fluid–solid coupling [2], artistic control [39] and GPU acceleration [9]. However, user interaction, which is essential for interactive applications, is rarely investigated in fluid simulation researches. As a result, current simulation techniques suffer from poor stability when user interaction is involved.

2.2 Boiling simulation

Boiling is a process in which rapid vaporization of liquid occurs when the liquid is heated to its boiling point. During the heating process, the fluid particles gain more and more momentum, and some of them manage to escape from the surface of the liquid as a vapor, hence displaying chaotic behavior.

Eulerian methods. Mihalef et al. [23] use an Eulerian method for physics-based boiling simulation. Their method extends coupled level set and volume of fluid (CLSVOF) [34] with temperature and mass transfer mechanisms and obtains visually rich animations. Kim and Carlson [13] proposed a grid-based modular approach, where boiling was treated as a separate module from the fluid solver. The modular design enables simple integration and high efficiency but also makes it challenging to implement detailed liquid–air interaction.

Lagrangian methods. In 2005, Müller et al. [28] proposed a particle-based technique to simulate air–water interaction by generating air particles on the fly where needed. To simulate boiling water, they randomly choose cavitation sites at the bottom of the container and transform water particles into air particles when the boiling point is reached. Artificial buoyancy is introduced to push bubbles to the fluid surface. Prakash et al. [31] use discrete particles to simulate bubbles for animating boiling effects. Their method focuses on animating the detail in the boiling process. They simulate the change in bubble scale depending on the local superheat of the fluid and create jets of steam particles when bubbles burst at the free surface. Unlike previous works, Gu et al. [11] simulate vapor bubbles rather than air bubbles. Vapor bubbles can condense and merge with the surrounding liquids, while air bubbles cannot. They focus on simulating the process of bubble formation, deformation and condensation. Li and Xiao [16] presented a boiling simulation framework based on position-based fluids (PBF) [19]. They proposed a simplified mass transfer model and established a more elaborate heat transfer model.

Previous researches focused on capturing the subtle details in the boiling process. Liquids are transformed into bubbles on caviation sites when the boiling point is reached and the motions of bubbles are updated to model the fluid motion indirectly. However, the acid dilution experiment involves extremely chaotic motion that can be difficult to simulate with previous methods. With bubble-based boiling simulation methods, bubbles get removed when they reached the fluid surface, and the system cannot gain enough momentum to capture the dramatic variation observed in violent boiling.

2.3 Interactive simulation

With the rapid improvements in information technology, interactive simulation has become a powerful tool and is widely used for education and training. Many people favor virtual experiments due to their safety and convenience. Maier and Klinker [22] designed the augmented chemical reaction application to visualize 3D chemical structures and simulate molecular dynamics. Luo et al. [18] developed a multi-channel user interface for mixed reality experiments. [5]. In recent years, position-based simulation methods have been widely applied in surgical training. Xiao et al. [37] proposed an interactive physics-based VR simulation framework for neonatal endotracheal intubation (ETI), where PBD was used to simulate soft tissues, rigid bones and fluids at interactive rates. Pan et al. [30] provided a VR medical training simulator based on PBD for cholecystectomy. Their work focuses on improving the realism of the fat tissue electrocautery and the liver–gallbladder separation.

3 Background

The fluid solver in our system follows position-based fluids (PBF) [19], a particle-based fluid simulation method targeted for real-time applications. PBF uses the smoothed particle hydrodynamics (SPH) [24,25] formalism for spatial discretization. More specifically, a scalar quantity A_i at particle i is approximately computed by a weighted sum of

contributions from all neighboring particles j:

$$A_i = \sum_j \frac{m_j}{\rho_j} A_j W(\mathbf{x}_i - \mathbf{x}_j, h), \tag{1}$$

where m_j is the particle mass, ρ_j the particle density, $\mathbf{x}_j$ the particle position and W the kernel function with support radius h.

To enforce constant fluid density, PBF solves a system of nonlinear constraints, with one constraint per particle:

$$C_i(\mathbf{x}_1, \cdots, \mathbf{x}_n) = \frac{\rho_i}{\rho_0} - 1 \leq 0, \tag{2}$$

where n is the number of particles, ρ_0 the rest density and ρ_i the density at particle i which can be calculated using Eq. 1.

For more details on particle-based fluid simulation, we refer the readers to the tutorials on SPH [14] and PBD [4].

4 Boiling simulation

4.1 Diffusion

The diffusion equation describes how the heat or concentration gets distributed in a fluid. The change in an attribute $A(\mathbf{x}, t)$ due to diffusion can be calculated as [28]:

$$\frac{\partial A(\mathbf{x}, t)}{\partial t} = D\nabla^2 A(\mathbf{x}, t), \tag{3}$$

where D is the diffusion coefficient for attribute A. Using SPH discretization, the evolution of attributes A_i at particle i can be computed as:

$$\frac{\partial A_i}{\partial t} = D \sum_j \frac{m_j}{\rho_j} (A_j - A_i) \nabla^2 W(\mathbf{x}_i - \mathbf{x}_j, h). \tag{4}$$

The problem with Eq. 4 is that the Laplacian of the kernel function $\nabla^2 W$ changes its sign inside the support radius, and it is sensitive to particle disorder [14]. Therefore, we use a modified diffusion equation proposed by Cummins and Rudman [6] that only involves first derivatives:

$$\frac{\partial A_i}{\partial t} = D \sum_j \frac{m_j}{\rho_j} (A_i - A_j) \left(\frac{4\rho_i}{\rho_i + \rho_j} \right) \frac{\mathbf{x}_{ij} \cdot \nabla W(\mathbf{x}_{ij}, h)}{|\mathbf{x}_{ij}|^2 + 0.01h^2}, \tag{5}$$

which is an approximation of Eq. 4 and prevents the numerical oscillation problem by removing the second-order derivative of the kernel function.

4.2 Heat of solution

The heat of solution describes the quantity of heat evolved or absorbed in the dissolution of a substance. For concentrated sulfuric acid, a large quantity of heat is liberated when mixing with water. If a small amount of water is added to a large volume of a strong acid, the dissolution may produce a sufficiently large ΔT that the water may boil violently and splash from the container, resulting in a safety hazard. The correct way to dilute concentrated sulfuric acid is by adding acid to water instead. Since water has a higher heat capacity than acid, the generated heat quickly dissipates into a large volume of water, preventing the system from overheating.

To simulate the boiling behavior in sulfuric acid dilution, we need to calculate the heat of solution quantitatively. More specifically, for each pair of neighboring particles i and j, given their concentration before and after diffusion, we want to obtain the amount of heat q_{ij} generated during dilution.

The heat of solution q_{ij} is defined as the change of enthalpy during the dissolution. Let $H(c)$ be the enthalpy contained by a particle with concentration c. Given two neighboring particles i and j and their concentration c_i and c_j, we first obtain the change in concentration Δc using Eq. 5 and then calculate q_{ij} as the change of enthalpy:

$$q_{ij} = H(c_i) + H(c_j) - H(c_i + \Delta c) - H(c_j - \Delta c). \tag{6}$$

A particle with concentration c and mass m can be seen as the result of diluting $n_a(c) = \frac{mc}{M(H_2SO_4)}$ moles of sulfuric acid with $n_w(c) = \frac{m(1-c)}{M(H_2O)}$ moles of water, where $M(H_2SO_4) = 98$ g/mol and $M(H_2O) = 18$ g/mol are the molar mass of sulfuric acid and water, respectively. Let $h(c)$ be the heat of solution during this process and h_0 be the enthalpy contained by one mole of pure sulfuric acid. Since change of enthalpy equals to heat of solution, we have:

$$H(c) = n_a(c)h_0 - h(c). \tag{7}$$

Putting Eqs. 6 and 7 together gives:

$$q_{ij} = h(c_i + \Delta c) + h(c_j - \Delta c) - h(c_i) - h(c_j). \tag{8}$$

Julius Thomsen was the first person to make comprehensive measurements on heat of solution for sulfuric acid [15,35]. Thomsen deduced a hyperbolic empirical equation to fit his results from experiment:

$$Q(k) = \frac{17860 \times k}{k + 1.7983}, \tag{9}$$

where $Q(k)$ is the heat evolved upon mixing 1 mole of sulfuric acid with k moles of water.

The relationship between $Q(k)$ and $h(c)$ is straightforward. $h(c)$ is the heat evolved upon mixing $n_a(c)$ moles of sulfuric acid with $n_w(c)$ moles of water and hence can be calculated using $Q(k)$ as:

$$h(c) = n_a(c)Q\left(\frac{n_w(c)}{n_a(c)}\right). \tag{10}$$

By plugging Eqs. 9 and 10 into Eq. 8, we can quantitatively calculate the amount of heat q_{ij} evolved from diluting particles i with j. Note that we do not distribute the heat of solution evenly between particles i and j. Instead, more heat is distributed to the particle with lower acid concentration:

$$q_i = q_{ij} \cdot \left(\frac{c_j}{c_i + c_j + \epsilon}\right) \tag{11}$$

$$q_j = q_{ij} \cdot \left(\frac{c_i}{c_i + c_j + \epsilon}\right), \tag{12}$$

where ϵ is a small constant used to avoid singularities when the denominator is close to zero. Finally, we introduce a user-defined factor k_d to control the influence of solution heat on temperature:

$$\Delta T_i^{\text{dilution}} = k_d q_i. \tag{13}$$

The imbalanced distribution of heat models the higher volumetric heat capacity of water and captures the different phenomenon between adding water to acid and adding acid to water. When water is added to the acid, most of the heat is distributed to the water, leading to violent boiling. On the other hand, when acid is added to water, the heat is distributed among the large volume of water, thus preventing the system from overheating.

4.3 Anti-viscosity

Boiling is a common phenomenon in our life. Previous researches [11,16,28,31] proposed various methods to model the boiling phenomenon by simulating bubble dynamics during the process. While bubbles are important for displaying subtle details, they introduce extra modeling complexity and require lots of computational resources, which is undesirable for real-time applications.

In this work, we propose *anti-viscosity*, a bubble-free method to simulate the violent boiling process with low performance overhead. Before describing the details of our *anti-viscosity* method, first we provide a brief overview on viscosity. Viscosity is responsible for introducing energy dissipation and enforcing coherent motion on fluid particles. In the Navier–Stokes equations, viscosity force is determined as:

$$\mathbf{f}^{\text{viscosity}} = \mu \nabla^2 \mathbf{v}. \tag{14}$$

By applying the SPH discretization to the viscosity term, the viscosity force on particle i is:

$$\mathbf{f}_i^{\text{viscosity}} = \mu \sum_j \frac{m_j}{\rho_j}(\mathbf{v}_j - \mathbf{v}_i)\nabla^2 W(\mathbf{x}_i - \mathbf{x}_j, h), \tag{15}$$

where μ denotes viscosity coefficient, a positive value to introduce energy dissipation.

Due to the same reason as we described in Sect. 4.1, the second derivative of the kernel function is problematic. Schechter et al. [32] pointed out that using a full treatment of physical viscosity along with non-physical coefficients tuned to stabilize the simulation is overkill. Therefore, they proposed XSPH viscosity, a simplified artificial viscosity term defined as:

$$\mathbf{f}_i^{\text{viscosity}} = \mu \sum_j \frac{m_j}{\rho_j}(\mathbf{v}_j - \mathbf{v}_i) W(\mathbf{x}_i - \mathbf{x}_j, h), \tag{16}$$

where the Laplacian term is removed compared to Eq. 15. This simplification makes viscosity much easier to tune and achieves better visual results.

In XSPH viscosity, μ is a constant parameter with non-negative value. However, in our model, a viscosity coefficient μ_i is calculated for each particle i depending on the particle temperature T_i as:

$$\mu_i = \begin{cases} \mu_0 k_v\left(\frac{T_B - T_i}{T_B}\right) & T_i \geq T_B \\ \mu_0 & T_i < T_B \end{cases}, \tag{17}$$

where μ_0 is the default viscosity coefficient, k_v is a user-defined parameter to control the boiling intensity and T_B is the boiling point. We adopt this varying viscosity approach where the viscosity coefficient of a particle with high temperature is scaled linearly with its temperature. The viscosity value becomes negative for high-temperature particles, hence the name anti-viscosity.

Normally, viscosity is used to introduce energy dissipation and enforce coherent motion between fluid particles. In our model, anti-viscosity works the opposite way and injects momentum into high-temperature particles, and reproduces the chaotic behavior observed in violent boiling. Different from previous approaches, our approach does not model bubbles and works directly on liquid particles. This modification eliminates the complexity behind the bubble dynamics, achieves better performance and obtains better controllability over the fluid motion. These properties make our method especially appealing for interactive applications. Figure 2 shows the increasingly violent boiling process when pouring water into concentrated sulfuric acid.

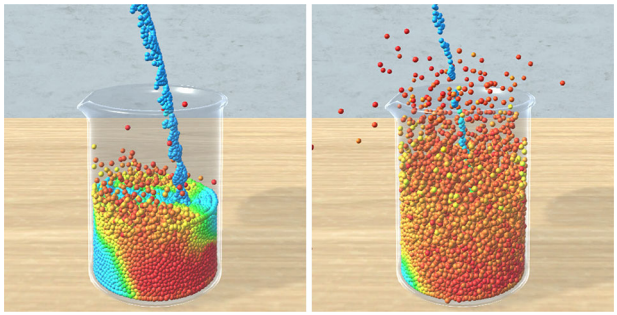


Fig. 2 The boiling becomes more and more violent as the water being poured into concentrated sulfuric acid. Particles are colored by temperature using a hot-to-cold color map

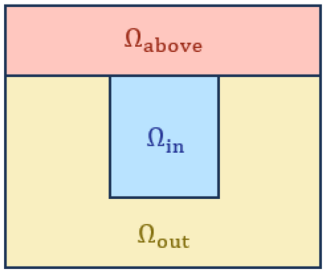


Fig. 3 A simplified 2D example of flask constraint. For each flask constraint, the simulation domain is divided into Ω_{in}, Ω_{out} and Ω_{above}. The regions are used for penetration detection and projection

5 Flask constraint

In particle-based fluid simulation, it is common to require that particles travel less than their radius in a single time step. Otherwise, collisions may be missed and particles may penetrate through their boundaries. This requirement is easy to meet for non-interactive simulations. However, the movement of solid particles is controlled by user interaction in our case. Particles can move with arbitrary velocity and result in severe particle penetration and unpleasant user experiences.

There are some ways to mitigate these issues, such as creating extra collision shape [37], restricting the maximum particle velocity [21], using one-sided collision normal [20]. These methods bring extra implementation complexity and performance overhead but still suffer from penetration when user-controlled objects are moving too fast. Even with mesh-based continuous collision detection, it is still possible for particles to tunnel if they are pushed through the triangle mesh by constraints during the constraint solving step [29].

We introduce a new constraint called *flask constraint* to address the particle penetration issue for interactive applications. Flask is a standard fluid container in chemistry experiments. Most of the flasks are axisymmetric or close to axisymmetric. The key idea of our approach is to use radius lookup tables to represent axisymmetric shapes. In this way, we can use an analytic expression to represent flasks' shapes and easily determine whether a particle penetrates the container during a time step. When a fluid particle penetrates these shapes, a flask constraint is generated and projects the particle back to a valid position inside the container. By using optimized expressions to represent the shape of the container, our method makes it much easier to implement robust particle penetration handling compared to mesh-based or particle-based methods.

Here we use a simplified 2D example to demonstrate our method. As shown in Fig. 3, the simulation domain is divided by the flask into Ω_{in} (inside), Ω_{above} (above) and Ω_{out} (outside) regions. Let $\mathbf{p_i}(x_i, y_i)$ be the center of flask i, and let l_i and w_i be the length and width of the flask. Now we can analytically express the three regions as:

$$\Omega_{\text{in}} = \{\mathbf{p}(x, y) \mid |x - x_i| < \frac{l_i}{2} \wedge |y - y_i| < \frac{w_i}{2}\} \quad (18)$$

$$\Omega_{\text{above}} = \{\mathbf{p}(x, y) \mid y \geq y_i + \frac{w_i}{2}\} \quad (19)$$

$$\Omega_{\text{out}} = \Omega_{\text{in}}^{\text{C}} \cap \Omega_{\text{above}}^{\text{C}}. \quad (20)$$

Given initial position $\mathbf{x}_i$ and predicted position $\mathbf{x}_i^*$ for particle i, if $\mathbf{x}_i \in \Omega_{\text{in}}$ and $\mathbf{x}_i^* \in \Omega_{\text{out}}$, the particle has potential to penetrate through the flask, hence a flask constraint is generated between the particle and the flask. When solving flask constraint, the predicted position $\mathbf{x}_i^*$ is corrected back into region Ω_{in} by projecting the position to the closest valid point in each axis.

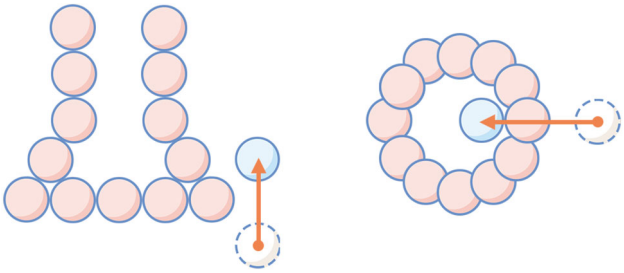
Since fluid containers can be freely translated and rotated by users in interactive applications, we store the local-to-world transformation matrix and world-to-local transformation matrix for each flask. When dealing with flask constraints, we first transform the particle position to the flask's local space. After the detection and projection, we transform the corrected position back into world space. In this way, we only need to deal with local coordinates, which significantly simplifies our implementation.

The extension to three-dimensional space is straightforward. To support any axisymmetric shape, we sample the radii of the shape on several horizontal slices during initialization and store these information into a lookup table. Let H_i be the height of flask i and $r_i(h)$ be the flask's radius on height h. Since we use local coordinates, the regions can be expressed as:

$$\Omega_{\text{in}} = \{\mathbf{p}(x, y, z) \mid x^2 + z^2 < r_i(y)^2 \wedge y < \frac{H_i}{2}\} \quad (21)$$

$$\Omega_{\text{above}} = \{\mathbf{p}(x, y, z) \mid y \geq \frac{H_i}{2}\} \quad (22)$$

$$\Omega_{\text{out}} = \Omega_{in}^{\text{C}} \cap \Omega_{\text{above}}^{\text{C}}. \quad (23)$$



(a) Vertical projection (b) Horizontal projection

Fig. 4 When a penetration occurs, the particle is projected on the vertical axis and horizontal plane separately

Figure 4 shows the process of solving flask constraint in 3D. Given a potentially penetrated particle, we first project it vertically to a proper position. Afterward, we use the corrected y coordinate to look up the flask's radius on current height $r(y)$ and project the particle towards the center on the horizontal plane.

Note that during the projection, we did not project the particle onto the exact border between Ω_{in} and Ω_{out}. Instead, we move the particles slightly inwards to avoid being pushed out by the boundary particles repeatedly. We use a parameter d_{margin} to control this margin. If d_{margin} is too small, particles may stay out of the flask after projection due to numerical error. If d_{margin} is too large, projections may result in drastic changes in particle position. A value of $d_{\text{margin}} = 0.001h$ works well for all our examples.

Flask constraint is easy to integrate and has little performance overhead. It is important to note that flask constraint is not a replacement for conventional collision handling methods. Instead, flask constraint is a safeguard mechanism used alongside traditional collision handling methods to avoid particle penetration during extreme movement. When there is no particle penetration, flask constraint will not be generated and thus have no influence on the physical accuracy of the fluid simulation. Though it may introduce particle compression when penetration occurs, the fluid will recover soon in several frames. Our method can provide robust user interaction and avoid particle penetration, as demonstrated in Fig. 5.

6 Implementation

Based on our proposed methods, we build our VR application using unity engine and virtual reality toolkit (VRTK) [8]. The fluid solver is implemented with compute shaders for parallel execution on GPU. For real-time fluid rendering, we compute particle anisotropy using the method of Yu and Turk [38] and reconstruct the surface using screen space filtering by Van der Laan et al. [36].

The simulation loop in our system is outlined in Algorithm 1. Compared with the simulation loop of position-based fluids, we introduced dilution simulation and flask constraints and integrated several optimizations.

Dilution simulation To allow users to move objects around in a VR environment, lines (1)–(5) update solid particles' positions so that they move along with the rigid bodies. To simulate viscosity between fluid and solid particles, we also calculate the velocity of solid particles. Line (23) generates and solves the flask constraints to avoid particle penetration. The temperature T_i and concentration c_i are updated by diffusion and heat of solution in Line (26). Note that the temperatures of solid particles are a fixed value to ensure liquid cools down to ambient temperature after heating. Line (27) applies viscosity or anti-viscosity to particles depending on their temperature.

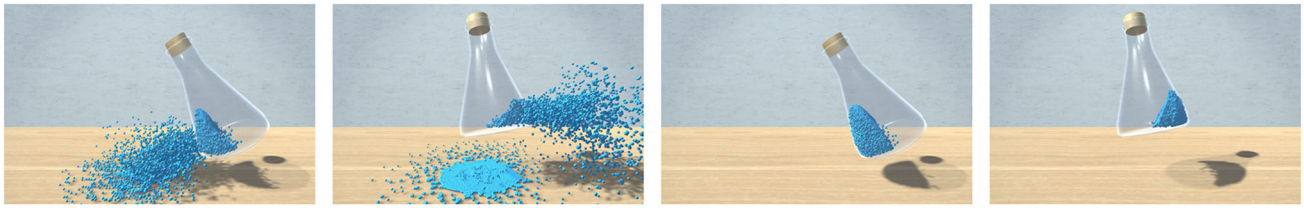
Optimizations Lines (6)–(13) use the substep optimization proposed by Macklin et al. [21], which improves incompressibility and reduces damping and instabilities. Following the hash implementation by Green [10], we also reorder particles in every n_{reorder} frames to improve the memory coherence on GPU. Line (20) applies under-relaxation [20] for density constraints using a user-defined parameter ω, where $0 < \omega \leq 1$. This under-relaxation technique helps alleviate the particle oscillation problem caused by slow convergence when the number of iterations is relatively small. Line (24) enforces a maximum particle velocity magnitude, which reduces particle tunneling and improves stability. Line (28) applies internal forces, including vorticity confinement [19], adhesion and cohesion forces [1].

7 Results

In this section, we demonstrate the results of our proposed methods in various simulated scenes. All examples are run on a laptop with a 6-core 2.6GHz CPU, an NVIDIA GeForce RTX 2060 graphics card and 16 GB of RAM.

Moving flask Our flask constraint deals with particle penetration issue that arises with rapid object movement. Figure 5 shows the different results of moving flasks with and without flask constraints. Density constraints and contact constraints are not designed to handle dramatic change in position and suffer from severe particle penetration. With help from flask constraints, the system can reliably avoid particle penetration.

Single dilution In Fig. 1 we demonstrate the simulation result for concentrated sulfuric acid dilution. In this example, distilled water is poured into concentrated sulfuric acid. A few seconds after pouring, the acid heats up rapidly and presents increasingly violent boiling behavior. Eventually, the acid cools down and gradually stabilizes. Although we do not involve bubbles in our calculation explicitly, anti-viscosity can still create empty spaces inside the liquid which is visually similar to bubbles.



(a) (b) (c) (d)

Fig. 5 (a)(b) Density constraint and contact constraint are not designed to cope with extreme movement, thus suffering from severe penetration. (c)(d) By integrating our flask constraint, the system can reliably avoid particle penetration under arbitrary movement

Algorithm 1 Simulation Loop

for all solid particles j **do**
  update position $\mathbf{x}_j^*$ according to user interaction
  calculate velocity $\mathbf{v}_j \leftarrow \mathbf{x}_j^* - \mathbf{x}_j$
  update position $\mathbf{x}_j \leftarrow \mathbf{x}_j^*$
end for
for all fluid particles i **do**
  predict position $\mathbf{x}_i^* \leftarrow \mathbf{x}_i + \Delta t_f \mathbf{v}_i + \Delta t_f^2 m^{-1} \mathbf{f}_{ext}(\mathbf{x}_i)$
end for
for all fluid particles i **do**
  find neighboring particles $N_i(\mathbf{x}_i^*)$
  reorder particles in every $n_{reorder}$ frames
end for
$\Delta t_s \leftarrow \frac{\Delta t_f}{n_{steps}}$
while $n < n_{steps}$ **do**
  for all fluid particles i **do**
    predict position $\mathbf{x}_i^* \leftarrow \mathbf{x}_i + \Delta t_s \mathbf{v}_i + \Delta t_s^2 m^{-1} \mathbf{f}_{ext}(\mathbf{x}_i)$
  end for
  while $iter < solverIterations$ **do**
    solve all density constraints for $\Delta \mathbf{x}$
    update $\mathbf{x}^* \leftarrow \mathbf{x}^* + \omega \Delta \mathbf{x}$
  end while
  for all fluid particles i **do**
    generate and solve flask constraints for corrected $\mathbf{x}^*$
    update velocity $\mathbf{v}_i \leftarrow min(\frac{1}{\Delta t_s}(\mathbf{x}_i^* - \mathbf{x}_i), \mathbf{v}_{max})$
    update position $\mathbf{x}_i \leftarrow \mathbf{x}_i^*$
    apply diffusion and heat of solution for T_i, c_i
    apply XSPH viscosity and anti-viscosity
    apply internal forces $\mathbf{f}^{adhesion}$, $\mathbf{f}^{cohesion}$, $\mathbf{f}^{vorticity}$
  end for
end while

Triple dilution In Fig. 6, we demonstrate that our method can produce different simulation results when mixing different liquids. The water remains stable in the first flask since adding water to water does not produce any heat. In the second flask, acid is added into water, which is the correct operation but still leads to heat generation and slight splashing. In the third flask, a large amount of heat is generated when water is added to the acid, leading to intensive boiling.

Tomato soup In Fig. 7, we demonstrate that our anti-viscosity model can also be applied to simulate other common boiling phenomena and allows easy control over the boiling phenomenon's intensity. In this example, a pot of boiling tomato soup is simulated, and the boiling intensity is controlled by the parameter k_v. When $k_v = 0$, the anti-viscosity effect is disabled, and the fluid surface remains stable. By modifying the value of k_v, we can easily achieve different levels of boiling intensity.

VR dilution We provide a real-time simulation of concentrated sulfuric acid dilution in a VR environment, as shown in Fig. 8. By using flask constraints, users can move objects around without worrying about particle penetration. By calculating heat of solution and employ anti-viscosity model, our system allows real-time simulation of violent boiling and can model different simulated results for different operations. By observing the dangerous consequences of incorrect operations, students can improve their safety awareness and prepare for real experiments in an interesting way.

Parameters and performance Table 1 summarizes the parameters used in our example scenes. μ_0 is the default viscosity coefficient. Ambient temperature T_a is used to adjust the default temperature for particles. Boiling point T_B is a temperature threshold above which boiling occurs. Heat generation coefficient k_d scales the heat generated during dilution. k_v controls the boiling intensity. D_T and D_c are the diffusion coefficient for temperature T and concentration c. The performance data are shown in Table 2. We use 3 substeps and 2 iterations for all example scenes. Comparing with the basic PBF solver, the average performance overhead of flask constraint and boiling simulation is 1.10% and 14.42%, respectively. The results show that our methods extend the capability and stability of PBF solver with little performance overhead.

8 Discussion and conclusion

We have proposed a comprehensive solution for simulating concentrated sulfuric acid dilution in real time. The heat of solution is calculated to simulate the exothermic process in diluting sulfuric acid. To capture the chaotic dynamics observed in violent boiling process, we propose the anti-viscosity model to inject momentum into particles with high temperatures. Additionally, we employ a constraint-based method to deal with the particle penetration issue arising

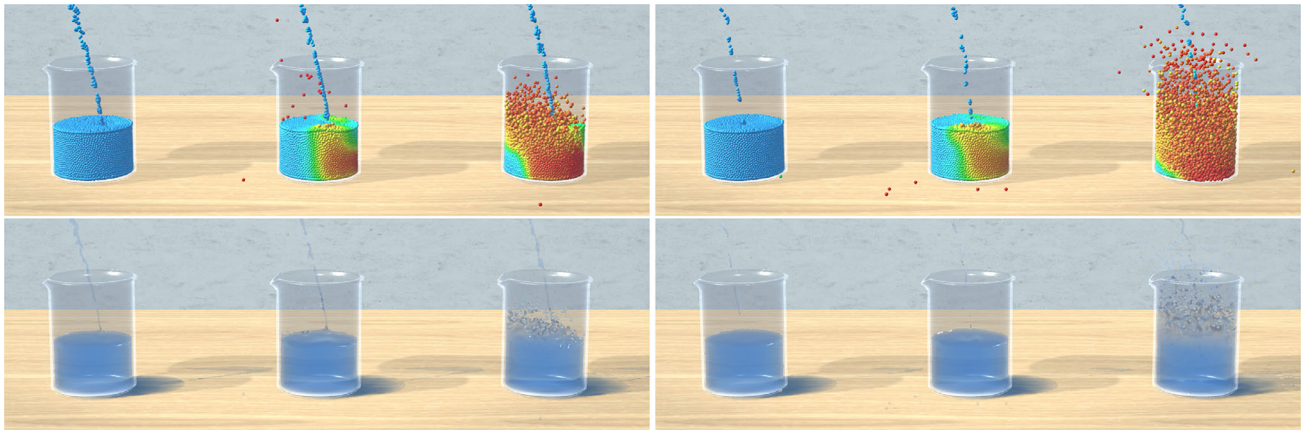


Fig. 6 We compare the result of adding water to water (left flask), adding acid to water (middle flask) and adding water to acid (right flask). Results show that our method can model different phenomena when mixing different liquids

Fig. 7 A pot of boiling tomato soup is simulated with different k_v. This shows that our method can model different levels of boiling intensity

Fig. 8 Diluting concentrated sulfuric acid in a VR environment. Our methods enable robust user interaction with fluids and provide real-time simulation results for both correct operation (pouring acid into water) and incorrect operation (pouring water into acid)

Table 1 Parameters used in demo scenes

	μ_0	T_a	T_B	k_d	k_v	D_T	D_c
Single dilution (Fig. 1)	0.3	20	100	0.017	1	0.01	0.007
Triple dilution (Fig. 6)	0.3	20	100	0.017	1	0.01	0.007
Tomato soup (Fig. 7)	0.3	110	100	–	0 / 0.75 / 0.9	–	–
VR dilution (Fig. 8)	0.3	20	100	0.017	1	0.01	0.007

Table 2 Performance data for demo scenes

Name	Moving flask (Fig. 5)	Single dilution (Fig. 1)	Triple dilution (Fig. 6)	Tomato soup (Fig. 7)
# fluid particles	10925	13894	39888	14880
# solid particles	51158	48044	60632	2341
Solve fluid (ms)	3.014	2.667	6.973	2.58
Solve boiling (ms)	0	0.380	1.003	0.379
Solve flasks (ms)	0.027	0.024	0.058	0.025
Rendering (ms)	0.722	1.256	1.776	4.271
Total (ms)	3.763	4.327	9.810	7.255

from user interaction. This work improves the capability and stability of position-based fluid solver and provides appealing solutions for integrating fluid simulation into interactive applications.

There are some limitations with our current methods. Since the anti-viscosity method introduce extra energy into the system, instabilities or even explosion can occur if parameters are not set properly. Second, we omit some differences between sulfuric acid and water in our simulation, such as different viscosity, color, density, heat capacity and boiling point. Considering these differences will provide an even more realistic simulation at higher performance overhead. It is a trade-off between accuracy and efficiency, which we will explore in the future. Additionally, our anti-viscosity model does not involve vaporization and condensation, which are also important phenomena in the dilution experiment. In our future work, we will investigate how to include phase changes in our model and improve visual fidelity.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s00371-021-02208-0.

Acknowledgements This work was supported by the National Key Research and Development Program of China (2018YFB1004902) and the National Natural Science Foundation of China (61772329, 61373085).

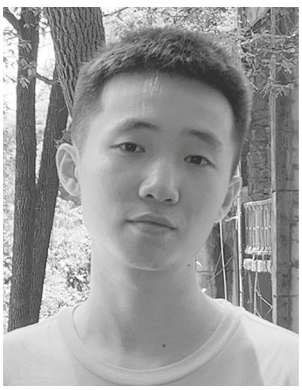
Declarations

Conflict of interest The authors have no conflicts of interest to declare that are relevant to the content of this article.

References

1. Akinci, N., Akinci, G., Teschner, M.: Versatile surface tension and adhesion for SPH fluids. ACM Trans. Graph. (TOG) **32**(6), 1–8 (2013)
2. Akinci, N., Ihmsen, M., Akinci, G., Solenthaler, B., Teschner, M.: Versatile rigid-fluid coupling for incompressible SPH. ACM Trans. Graph. (TOG) **31**(4), 1–8 (2012)
3. Bender, J., Koschier, D.: Divergence-free SPH for incompressible and viscous fluids. IEEE Trans. Vis. Comput. Graph. **23**(3), 1193–1206 (2016)
4. Bender, J., Müller, M., Macklin, M.: A survey on position based dynamics, 2017. In: Proceedings of the European Association for Computer Graphics: Tutorials, EG '17. Eurographics Association, Goslar, DEU (2017). https://doi.org/10.2312/egt.20171034
5. Chowdhury, T.I., Ferdous, S.M.S., Quarles, J.: VR disability simulation reduces implicit bias towards persons with disabilities. IEEE Trans. Vis. Comput. Graph. (2019)
6. Cummins, S.J., Rudman, M.: An SPH projection method. J. Comput. Phys. **152**(2), 584–607 (1999)
7. Desbrun, M., Gascuel, M.P.: Smoothed particles: A new paradigm for animating highly deformable bodies. In: Computer Animation and Simulation'96, pp. 61–76. Springer (1996)
8. Extend Reality Ltd: VRTK - virtual reality toolkit. https://www.vrtk.io/
9. Goswami, P., Schlegel, P., Solenthaler, B., Pajarola, R.: Interactive SPH simulation and rendering on the GPU. In: Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '10, p. 55–64. Eurographics Association, Goslar, DEU (2010)
10. Green, S.: Particle simulation using cuda. NVIDIA whitepaper **6**, 121–128 (2010)
11. Gu, Y., Yang, Y.H.: Physics based boiling bubble simulation. In: SIGGRAPH ASIA 2016 Technical Briefs, SA '16. Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/3005358.3005385
12. Ihmsen, M., Cornelis, J., Solenthaler, B., Horvath, C., Teschner, M.: Implicit incompressible SPH. IEEE Trans. Vis. Comput. Graph. **20**(3), 426–435 (2013)
13. Kim, T., Carlson, M.: A simple boiling module. In: Symposium on Computer Animation: Proceedings of the 2007 ACM

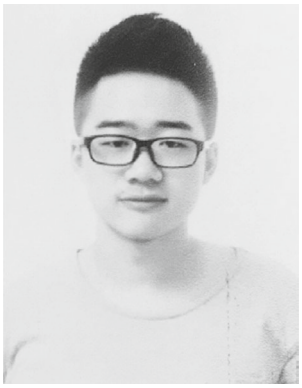
SIGGRAPH/Eurographics Symposium on Computer Animation, vol. 2, pp. 27–34. Citeseer (2007)
14. Koschier, D., Bender, J., Solenthaler, B., Teschner, M.: Smoothed Particle Hydrodynamics Techniques for the Physics Based Simulation of Fluids and Solids. In: W. Jakob, E. Puppo (eds.) Eurographics 2019 - Tutorials. The Eurographics Association (2019). https://doi.org/10.2312/egt.20191035
15. Leenson, I.: Sulfuric acid and water: paradoxes of dilution. J. Chem. Educ. **81**(7), 991 (2004)
16. Li, Z., Xiao, S.: Boiling simulation of position based fluid. In: Proceedings of the 4th International Conference on Virtual Reality, pp. 142–146 (2018)
17. Lucy, L.B.: A numerical approach to the testing of the fission hypothesis. Astron. J. **82**, 1013–1024 (1977)
18. Luo, T., Zhang, M., Pan, Z., Li, Z., Cai, N., Miao, J., Chen, Y., Xu, M.: Dream-experiment: a MR user interface with natural multi-channel interaction for virtual experiments. IEEE Trans. Vis. Comput. Graph. **26**(12), 3524–3534 (2020)
19. Macklin, M., Müller, M.: Position based fluids. ACM Trans. Graph. (TOG) **32**(4), 1–12 (2013)
20. Macklin, M., Müller, M., Chentanez, N., Kim, T.Y.: Unified particle physics for real-time applications. ACM Trans. Graph. (TOG) **33**(4), 1–12 (2014)
21. Macklin, M., Storey, K., Lu, M., Terdiman, P., Chentanez, N., Jeschke, S., Müller, M.: Small steps in physics simulation. In: Proceedings of the 18th Annual ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 1–7 (2019)
22. Maier, P., Klinker, G.: Augmented chemical reactions: an augmented reality tool to support chemistry teaching. In: 2013 2nd Experiment@ International Conference (exp. at'13), pp. 164–165. IEEE (2013)
23. Mihalef, V., Unlusu, B., Metaxas, D., Sussman, M., Hussaini, M.Y.: Physics based boiling simulation. In: Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 317–324 (2006)
24. Monaghan, J.J.: Smoothed particle hydrodynamics. Ann. Rev. Astron. Astrophys. **30**(1), 543–574 (1992)
25. Monaghan, J.J.: Simulating free surface flows with SPH. J. Comput. Phys. **110**(2), 399–406 (1994)
26. Müller, M., Charypar, D., Gross, M.H.: Particle-based fluid simulation for interactive applications. In: Symposium on Computer animation, pp. 154–159 (2003)
27. Müller, M., Heidelberger, B., Hennix, M., Ratcliff, J.: Position based dynamics. J. Vis. Commun. Image Represent. **18**(2), 109–118 (2007)
28. Müller, M., Solenthaler, B., Keiser, R., Gross, M.: Particle-based fluid–fluid interaction. In: Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 237–244 (2005)
29. NVIDIA: Manual – NVIDIA flex 1.2.0 documentation. https://gameworksdocs.nvidia.com/FleX/1.2/lib_docs/manual.html
30. Pan, J., Zhang, L., Yu, P., Shen, Y., Wang, H., Hao, H., Qin, H.: Real-time VR simulation of laparoscopic cholecystectomy based on parallel position-based dynamics in GPU. In: 2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR), pp. 548–556. IEEE (2020)
31. Prakash, M., Cleary, P.W., Pyo, S.H., Woolard, F.: A new approach to boiling simulation using a discrete particle based method. Comput. Graph. **53**, 118–126 (2015)
32. Schechter, H., Bridson, R.: Ghost SPH for animating water. ACM Trans. Graph. (TOG) **31**(4), 1–8 (2012)
33. Solenthaler, B., Pajarola, R.: Predictive-corrective incompressible SPH. In: ACM SIGGRAPH 2009 Papers, SIGGRAPH '09. Association for Computing Machinery, New York, NY, USA (2009). https://doi.org/10.1145/1576246.1531346
34. Sussman, M.: A second order coupled level set and volume-of-fluid method for computing growth and collapse of vapor bubbles. J. Comput. Phys. **187**(1), 110–136 (2003)
35. Thomsen, J.: Thermochemische untersuchungen: bd. Metalloide. 1882, vol. 2. JA Barth (1882)
36. van der Laan, W.J., Green, S., Sainz, M.: Screen space fluid rendering with curvature flow. In: Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games, I3D '09, p. 91–98. Association for Computing Machinery, New York, NY, USA (2009)
37. Xiao, X., Zhao, S., Meng, Y., Soghier, L., Zhang, X., Hahn, J.: A physics-based virtual reality simulation framework for neonatal endotracheal intubation. In: 2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR), pp. 557–565. IEEE (2020)
38. Yu, J., Turk, G.: Reconstructing surfaces of particle-based fluids using anisotropic kernels. ACM Trans. Graph. (TOG) **32**(1), 1–12 (2013)
39. Zhang, S., Yang, X., Wu, Z., Liu, H.: Position-based fluid control. In: Proceedings of the 19th Symposium on Interactive 3D Graphics and Games, pp. 61–68 (2015)



Wentao Chen received the B.S. degree in software engineering from Shanghai Jiao Tong University in 2019. He is currently pursuing the M.S. degree at Digital ART Lab of School of Software at Shanghai Jiao Tong University. His research interests include real-time physics simulation and virtual reality.

Tian Sang received her B.S. degree in software engineering from Nanjing University in 2019. Currently, she is a Master student at Digital ART Lab of School of Software at Shanghai Jiao Tong University. Her research interests lie in computer graphics and virtual reality.

Yitian Ma is an undergraduate student in the Department of Computer Science at Shanghai Jiao Tong University. She is also an intern at Digital ART Lab of School of Software at Shanghai Jiao Tong University. Her research interests lie in computer graphics, including rendering and physics-based simulation.



Qian Chen received his B.S. degree in computer science from Wuhan University in 2019. He is currently a Ph.D. student at Digital ART Lab of School of Software at Shanghai Jiao Tong University. His research interests lie in deep learning, fluid simulation and computer graphics.

Yuwei Xiao is a master student at Digital ART Lab of School of Software at Shanghai Jiao Tong University. His research interests include large-scale fluid simulation, high-performance computing and computer graphic-related topics.

Zhigeng Pan received his Ph.D. degree in 1993 from Zhejiang University. He was the director of Digital Media and HCI Research Center in Hangzhou Normal University. Now, he is the Dean of School of Artificial Intelligence, Nanjing University of Information Science & Technology. His research interests include human–computer interaction, virtual reality, multimedia, and digital entertainment.

Xubo Yang received his Ph.D. degree in computer science from the State Key Lab of CAD and CG, Zhejiang University, in 1998. He is a full professor and the director of Digital ART Lab of School of Software at Shanghai Jiao Tong University. His research interests include computer graphics, virtual reality and human–computer interaction. He is the corresponding author of this paper.