3D Res (2018)9:4
https://doi.org/10.1007/s13319-018-0156-0

3DR EXPRESS

Physically-Based Rendering of Particle-Based Fluids with Light Transport Effects

Ali Beddiaf · **Mohamed Chaouki Babahenini**

Received: 5 December 2017 / Revised: 14 January 2018 / Accepted: 15 January 2018

Abstract Recent interactive rendering approaches aim to efficiently produce images. However, time constraints deeply affect their output accuracy and realism (many light phenomena are poorly or not supported at all). To remedy this issue, in this paper, we propose a physically-based fluid rendering approach. First, while state-of-the-art methods focus on isosurface rendering with only two refractions, our proposal (1) considers the fluid as a heterogeneous participating medium with refractive boundaries, and (2) supports both multiple refractions and scattering. Second, the proposed solution is fully particle-based in the sense that no particles transformation into a grid is required. This interesting feature makes it able to handle many particle types (water, bubble, foam, and sand). On top of that, a medium with different fluids (color, phase function, etc.) can also be rendered.

Keywords SPH simulation · Fluid visualization · Global illumination · Path tracing

A. Beddiaf (✉) · M. C. Babahenini
Department of Computer Science, LESIA Laboratory, University Mohamed Khider of Biskra, Biskra, Algeria
e-mail: alibeddiaf@gmail.com

M. C. Babahenini
e-mail: chaouki.babahenini@gmail.com

1 Introduction

In graphical animation domain, two major classes of physical simulation have been proposed to study fluid dynamics: (1) grid-based and (2) Lagrangian particle-based approaches. In the first category of approaches [9, 24], a Eulerian grid is utilized by the solver in order to store in every cell the fluid properties (e.g., density, velocity, pressure, etc.). Whereas, in the second category [25, 26], the fluid motion is determined by solving the particle set interaction forces. The result of this operation is a 3D point-set and other simulation variables.

In order to visualize the simulation output, the intuitive idea that comes to mind is to use an existing point-based rendering technique. Actually, point-based rendering has been extensively studied in numerous research works and many effective techniques are used today to render point-clouds. Among these techniques, we first cite the splatting [38] which consists of local surface reconstruction (each point is replaced by a circular or elliptical disk combined with a reconstruction filter). A second interesting visualization technique is Moving Least Squares (MLS) [1, 2] which approximates an analytic continuous surface from the point set.

We note that these point-based techniques are all surface-based [26, 31] that handle only the external part of the fluid and render it with a limited number of refractions, namely, one refraction at the front-

Published online: 03 February 2018

facing surface [20, 26, 31], two refractions at the front-facing and back-facing surfaces [34, 35], or at most four refractions [15]. In reality, the light rays may refract many times inside the liquid before leaving it, and this is explained by the TIR phenomenon (Total Internal Refraction). So, for realistic rendering, multiple refractions should be considered during ray tracing. In addition to that, in fact, the points produced by the simulation solver cannot be handled as surface elements. The points do not only represent a surface but also the internal volume (in which photometric and physical properties change from point to another).

In this paper, we propose a solution that adequates the volumetric path tracing to one of the well-known particle-based simulation methods, namely Smoothed Particle Hydrodynamics (SPH). This choice is explained by the fact that path tracing produces realistic results (that include global illumination effects) due to its physically based nature. In our proposal, we consider the fluid as a participating medium with refractive boundaries, and this with the aim to treat both the external (surface) and internal (volume) aspects.

The remainder of this paper is organized into six sections as follows. The next section reviews existing methods related to fluid simulation and visualization. Section 3 presents the background and preliminaries required in this paper. More specifically, Sect. 3 recalls the principles of light transport on participating media. Section 4 details the proposed solution while Sect. 5 presents the experimentation made on simulation data given by several solvers. Finally, Sect. 6 concludes the paper and gives some issues that need to be tackled.

2 Related Work

Smoothed Particle Hydrodynamics (SPH) was initially proposed with the aim of studying and simulating astrophysical problems [11]. Later, SPH technique has been used in graphics field to simulate the behavior of fire and gases [28]. Then, SPH has been used for animating deformable bodies [6]. In the fluid simulation field, Müller et al. [26] were the first to propose the use of SPH. In fact, Müller et al. have proposed a technique to render the SPH particles. In their approach, using the color field technique, the particles that belong to the surface are identified along with their normals. Then, the surface particles are whether directly rendered with a splatting technique [38], or after triangulation phase using Marching Cubes algorithm [23]. The resulting images of this approach are not realistic [26].

Points, as primitive of rendering, have attracted many research efforts and shown their efficiency. Nonetheless, particles cannot be considered as points because they do not have normals. As a consequence of that, classical point-based techniques fail to directly render particle-clouds. To solve this issue, an extension of the splatting technique has been proposed to render the isosurface of particles [5]. However, even this approach seems impractical since it is unknown how the density field is computed in it.

Most particle-based simulation data are rendered using the density information which is organized into a 3D grid. Each grid cell stores a weighting of the nearby particle density. The isosurface can be extracted from the density grid by polygonalization using Marching Cubes algorithm [23]. This algorithm is considered as a very memory-bandwidth consuming solution. The isosurface can also be rendered from the density grid (usually after smoothing/refinement phase) by using the level set method [8]. The latter is computationally expensive.

The density grid can also be used to render the isosurface by GPU-based raycasting approaches [12, 13, 15]. For volume rendering, the grid-based model can be rendered by volume ray casting. In such a method, the ray passing through the grid is sampled and the color of the samples are blended front-to-back [7].

Other methods do not transform particles into a grid, and handle them as basis functions (smoothing isotropic/anisotropic kernels). More specifically, using a ray marching approach, a scalar (distance) function is calculated at different positions by summing the contribution of the different nearby particle kernels. This way, an implicit surface is tracked. The first work in this category, which uses an isotropic kernel, has been presented by Zhu and Bridson [37]. A second work assigns an anisotropic kernel to each

particle after Principal Component Analysis (PCA) of the particle cloud [36], and this in order to handle sharp features of the fluid. As a third example, in [31], the particle-based simulation data have been used to directly render a smooth surface on screen space using curvature flow.

Physically-based rendering has the advantage of producing realistic images which contain global illumination effects (i.e., direct and indirect lighting are considered during the image computation). Regarding participating media, the Radiative Transfer Equation (RTE) is solved either by a volumetric version of the path tracing techniques [21, 22] or by photon mapping [17, 19]. Several recent papers have addressed the problem of rendering participating media with refractive boundaries [14, 33] by which we are interested in this work.

In the present paper, we avoid particles-to-grid transformation and use particles to directly track an implicit surface (using a distance function ray tracing approach, as previously explained). The objective is to handle the fluid as a heterogeneous participating medium with refractive boundaries. Because, in fact, the output of the physical simulation phase is not only the positions of particles but also their physical/photometric properties (like density which encloses absorption and scattering, albedo, phase function, etc.). Finally, the rendering is performed by adequating the volumetric path tracing to the SPH-based fluids.

3 Theoretical Background of Light Transport on Participating Media

The Rendering Equation (RE) has been generalized to Radiative Transfer Equation (RTE) to simulate light transport on participating media [19, 22] by path tracing and photon mapping. The RTE defines the radiance coming from a point x in the direction $\overrightarrow{\omega}$ as follows:

$$\begin{aligned}\frac{\delta L(x,\overrightarrow{\omega})}{\delta x} &= \alpha(x)L_e(x,\overrightarrow{\omega}) \\ &+ \sigma(x)\int_{\Omega} f(x,\overrightarrow{\omega}',\overrightarrow{\omega})L(x,\overrightarrow{\omega'})d\omega' \\ &- \alpha(x)L(x,\overrightarrow{\omega}) - \sigma(x)L(x,\overrightarrow{\omega})\end{aligned} \tag{1}$$

where L_e is the emitted radiance. α and σ are respectively the absorption and out-scattering coefficients. When the latter are constant, the medium is said to be homogenous, otherwise, it is heterogeneous. f is the phase function of the medium which is defined over $\Omega = 4\pi$. It is similar to the Bidirectional Reflectance Distribution Function (BRDF) on surfaces. f is often symmetric and depends only on the phase angle θ (between incidence and reflection directions). The medium is isotropic (or anisotropic) when the phase function is independent (or dependent respectively) on the phase angle.

We note that Eq. (1) has four radiance components: emission, in-scattering, out-scattering, and absorption. Usually, absorption and out-scattering terms are combined in one term called extinction:

$$\begin{aligned}k(x) &= \alpha(x) + \sigma(x) \\ k(x)L(x,\overrightarrow{\omega})\end{aligned} \tag{2}$$

where $k(x)$ is the extinction coefficient (also called density).

The integration of RTE along a segment $[x_0, x]$ of a ray passing through a medium gives:

$$\begin{aligned}L(x,\overrightarrow{\omega}) &= \int_{x_0}^{x} \tau(x',x)\alpha(x')L_e(x',\overrightarrow{\omega})dx' \\ &+ \int_{x_0}^{x} \tau(x',x)\sigma(x')\int_{\Omega} f(x',\overrightarrow{\omega}',\overrightarrow{\omega})L(x',\overrightarrow{\omega}')d\omega' dx' \\ &+ \tau(x_0,x)L(x_0,\overrightarrow{\omega})\end{aligned} \tag{3}$$

The main term in Eq. (3) is a double integral over two domains:

- Segment of the ray inside the medium.
- Sphere, where a phase function is defined at each point on the segment.

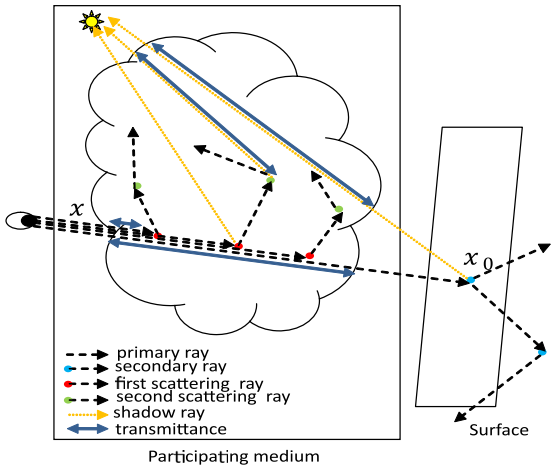

Fig. 1 Path tracing of participating media

$\tau(x', x)$ is the transmittance along the segment $[x', x]$ which decreases with the integrated density as follows:

$$\tau(x', x) = e^{-\int_{x'}^{x} k(\psi) d\psi} \tag{4}$$

As depicted in Fig. 1, the RTE can be solved with an unbiased stochastic approach like Monte Carlo path tracing (volumetric path tracing). In addition to ray-surface intersections computation and secondary rays generation (already considered in path tracing on surface-based scenes), a ray marching inside the medium is applied with scattering rays generation.

This ray marching is driven by random sampling of the ray segment according to the transmittance (integrated density) usually with importance sampling. The latter is better than the uniform sampling since the transmittance function is an exponential function of the integrated density (Fig. 2a). The integrated density is an increasing function of the ray samples (Fig. 2b). Therefore, for any solution of the RTE using path tracing, the transmittance (integrated density) calculation [i.e., integration of Eq. (4)] is the first question that has to be answered [18].

3.1 Density Integral Calculation for Participating Media

Due to their complexity, translucent material, and fuzzy shape, many objects in nature (like fog, smoke, skin, etc.) cannot be represented (and hence rendered) with classical polygonal models.

The output of the physical simulator is a set of sparse points that have physical and photometric properties like position, density, color, mass, volume, velocity, and phase function. The most important properties for the rendering are the position and density (which encloses absorption and scattering).

One of the most common techniques [9, 24] used to calculate the density integral along the ray passing through the participating medium is to first transform the particles density into a three-dimensional grid. Second, the ray is sampled into a set of points. Third, the density at each point is picked up using a trilinear interpolation of the surrounding cell. Finally, the integration is performed using numerical approaches like Woodcock tracking or Simpson integration.

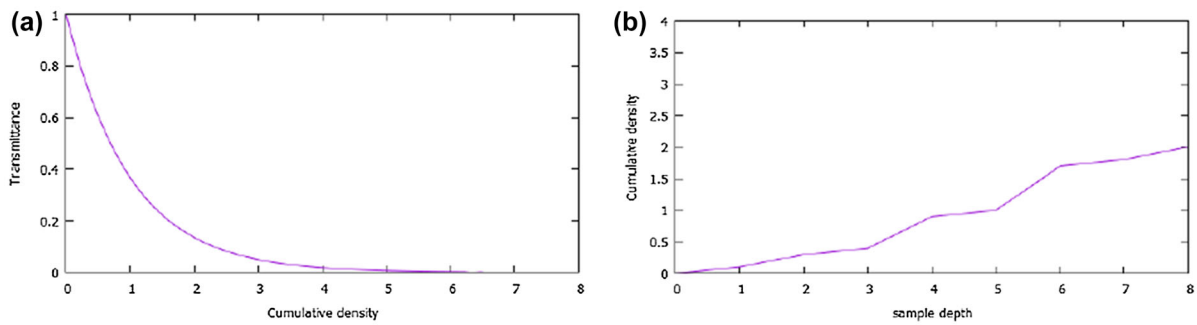

Fig. 2 Example of **a** transmittance function, **b** integrated density

4 Proposed Solution: Global Illumination for SPH-Based Fluids

In this section, we describe our solution for SPH-based fluids rendering. In more specific words, our approach adapts the volumetric path tracing for particle-based models and its operation is divided into two stages: (1) particle-based model of density integral calculation, and (2) fluid surface definition.

4.1 Particle-Based Model of Density Integral Calculation

As our approach considers heterogeneous participating media (i.e., particles with different densities), an efficient algorithm (Algorithm 1) has been proposed to calculate the density integral over the ray passing through overlapped particles (Fig. 3).

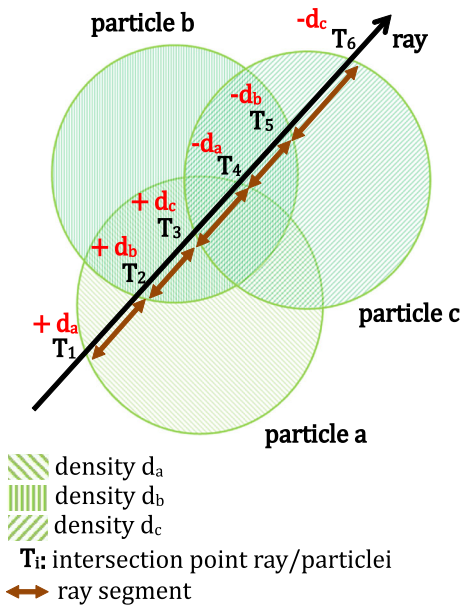

Fig. 3 Density integral calculation for a particle-based fluid

Algorithm 1: Density integral calculation.

Data: Particle Set P, Ray ray, Desired Density DD
Result: Position T
/*** Data setting ***/
struct Intersection{
float $tpos$;
float $density$;
int $particleID$;
Intersection(float t, float d, int id)
{
$tpos \longleftarrow t$;
$density \longleftarrow d$;
$particleID \longleftarrow id$;
}
}
vector<Intersection> IT;
int i,$counter$; float t1,t2;
bool $FOUND$;
float $TotalDensity$, $newDensity$, $difference$;
float $step$, $offset$, $TotalStep$, D, $scale$;
/*** Initialization ***/
$scale \longleftarrow 100.0$;
$TotalStep \longleftarrow 0.0$;
$TotalDensity \longleftarrow 0.0$;
$D \longleftarrow 0.0$;
$counter \longleftarrow 0$;
/*** Intersections computation ***/
foreach *element i of particle set P* **do**
  if *rayIntersectParticle(ray,P[i],t1, t2)* **then**
    IT.push(new Intersection($t1$,$P[i].density$,i) ;
    IT.push(new Intersection($t2$,-1*$P[i].density$,i)
    ;
QuickSort(IT);
/*** Integration of piecewise density function till getting the desired density ***/
$FOUND \longleftarrow false$;
for $j \leftarrow 0$ **to** $IT.size()$-1 **do**
  $newDensity \longleftarrow 0$;
  $step \longleftarrow IT[j+1].tpos - IT[j].tpos$;
  $D \longleftarrow D + IT[j].density$;
  if $IT[j].density > 0$ **then**
    $counter \longleftarrow counter + 1$;
  else
    $counter \longleftarrow counter - 1$;
  if $counter \neq 0$ **then**
    /* averaging the density in overlap regions */
    $newDensity \longleftarrow D/counter$;
  $TotalDensity \longleftarrow TotalDensity + newDensity * step * scale$;
  if $TotalDensity > DD$ **then**
    $difference \longleftarrow TotalDensity - DD$;
    $offset \longleftarrow difference/(newDensity * scale)$;
    $TotalStep \longleftarrow TotalStep + offset$;
    $T \longleftarrow TotalStep$;
    $FOUND \longleftarrow true$;
    break ;
  $TotalStep \longleftarrow TotalStep + step$;
/*** Result ***/
if $FOUND$ **then**
  Print("Desired Density found at", T) ;
else
  Print("Desired Density not found") ;

Once the desired density is randomly given as input to Algorithm 1, this latter produces as output the position T over the ray passing through the medium. In a nutshell, Algorithm 1 operates as follows. First, the ray-sphere intersections are determined and stored along with their respective densities in an array of structure. This array somehow represents a piecewise density function. Second, the integral of the density function is calculated over the ray segments until the desired density is reached.

Note that due to the fact that the overlapped particles belong to the same SPH-fluid, the density of the segments in the overlap regions is averaged. This choice can be changed if multiple SPH-fluids are considered.

In order to accelerate the first phase of Algorithm 1 (ray-sphere intersections computation), particles have been organized into two different spatial subdivisions; SAH KD-tree [32] and uniform grid. According to the obtained experimental results, the uniform grid clearly outperforms the KD-tree. Actually, this is obvious because SPH-particles are in general uniformly distributed in space and the traversal of a uniform grid needs less time compared to a KD-tree hierarchy.

Unlike grid-based approaches, our particle-based model of density integral calculation is more accurate because it is based on the original particles (not the transformed ones). This interesting characteristic gives more flexibility to the proposed model in the next rendering stages.

4.1.1 Multiple SPH Fluids Support

In this section, we extend our model of density integral calculation in order to handle several SPH-fluids with different properties (density, color, phase function, etc.). As previously mentioned, we chose to average the density in overlap regions. This is explained by the fact that the particles belong to the same fluid. Which is not the same case if a medium with multiple SPH-fluids has been considered (see Fig. 4).

The ambiguity issue of density integral calculation in the particle overlap regions can be resolved by the following three steps:

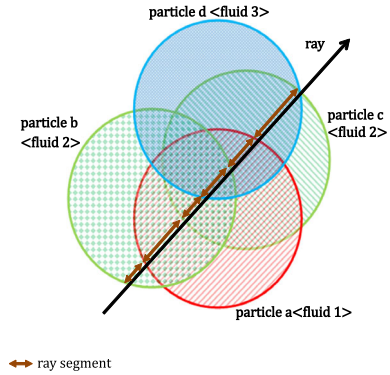

Fig. 4 Multiple SPH fluids

1. On each segment, the particles in question must be classified according to the fluid to which they belong.
2. Density of each class is averaged.
3. A class can be selected according to one of two strategies:
   (a) The lowest density.
   (b) The highest density.

4.2 Fluid Surface Definition for Participating Media

Fluids are translucent objects that have a surface which must be defined before any rendering operation. Actually, many events occur on the fluid surface (reflection and refraction), in addition to the events that happen inside the medium like absorption, emission, in-scattering, and out-scattering.

Particle-based approaches use a color field technique to identify the surface particles with their normals. Afterward, the visualization is achieved by splatting [26].

Grid-based approaches use the density grid to define a distance function in order to reconstruct an implicit surface by ray marching. Marching Cubes [23] is also usually used with grid-based simulation data. It gives a polygon soup, but with a costly price in memory, especially when a finer-meshed surface is targeted. The polygonal model could be rendered by a backward approach (projec-

tion/rasterization on the screen) or maybe raytraced on CPU or GPU.

In our work, we propose a ray tracing approach to both compute the density and render the surface with an unbiased technique (path tracing).

4.2.1 Convolution Surfaces

Convolution surfaces are a modeling approach for a specific type of surfaces (called soft objects) which cannot be represented easily with a mesh.

Starting from a point set, we define a kernel $f(x)$ with a radius $R1$ assigned to each point. These kernels may overlap each other. To find the surface's interface, we compute a distance function $D(x)$ that sums the different contributions of the nearby points. When this function reaches a threshold T, the surface is considered to be reached (Fig. 5).

$$D(x) - T = \sum f(x) - T = 0 \tag{5}$$

Equation 5 can be solved by Marching Cubes. However, we are interested rather in a ray marching approach in which the root can be found by a numerical iterative approach such as false position, secant, or bisection techniques.

We note that the larger the number of iterations, the accurate the result, but with extra computation time.

The kernel of convolution surfaces must be a continuous decreasing symmetric function. As a first example, let us consider the Jim Blinn's kernel defined by $f(r) = ae^{-br^2}$

The drawback of this Jim Blinn's model lies in the fact that the kernel is not bounded [all points contribute in $D(x)$] and the exponential term requires extra computational cost.

As a second example, let us consider the kernel of Zhu-Bridson [37] which avoids the exponential function:

$$f(r) = max(0, (1 - (r/b)^2)^3)$$

One of the classical examples of convolution surfaces is the metaball model [3] known by the following kernel:

$$f(r) = \begin{cases} a\left(1 - \dfrac{3r^2}{b^2}\right) & 0 \le r \le b/3 \\ \dfrac{3a}{2}\left(1 - \dfrac{r}{b}\right)^2 & \text{b/3} \le r \le b \\ 0 & \text{r} \ge b \end{cases}$$

Having the advantages of bounded kernel and low computation cost, the metaball model has been chosen, in this paper, to propose a rendering solution for SPH fluids.

Note that the value of the threshold T controls the resulting surface (Fig. 6):

- The smaller T, the coarser and voluminous the result will be.
- The larger T, the finer the result will be (but with loss of some surface elements).

4.2.2 Metaball Solution for SPH Particles

According to the conducted physical simulations of SPH fluids, each particle has a constant radius $R0$. Supposing that particles are widely spaced (no inter-particle overlap), a threshold $T0$ can be defined so that particles can be visualized as spheres with a radius $R0$ (see Eq. 6):

$$\begin{aligned} D(x) - T0 &= \sum f(x) - T0 \\ &= f(R0) - T0 = 0 \Rightarrow T0 = f(R0) \end{aligned} \tag{6}$$

The normal can be found by the gradient of the distance function $D(x)$ (using partial derivatives). As we work with a ray tracing approach, we estimate the normal as follows:

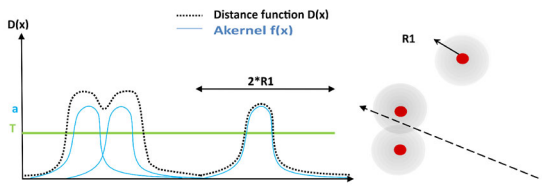

Fig. 5 Distance function of a ray traversing particles

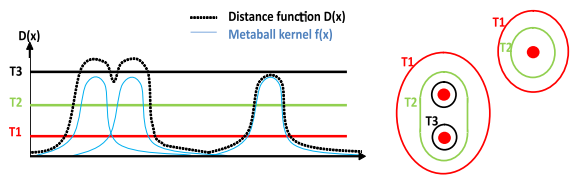

Fig. 6 Resulting surface of distance function with three different thresholds

$$Normal(x) = \frac{\sum f(||\overrightarrow{x - P_{i\cdot center}}||) * (\frac{\overrightarrow{x - P_{i\cdot center}}}{||\overrightarrow{x - P_{i\cdot center}}||})}{\sum f(||\overrightarrow{x - P_{i\cdot center}}||)}$$

In general, particles overlap each other, therefore, the use of the previous threshold yields a surface slightly coarser and smoother than spheres (and this, only in the overlap regions). Hence, the fluid final surface will appear too bumpy and will not look like a liquid (Fig. 7).

In order to make the fluid surface smoother and less bumpy, more particles from higher ring neighborhood must be considered during the normal computation. This idea stems from discrete curvatures estimation on triangular meshes [10] (Fig. 8).

Nevertheless, since particles are discrete and do not have a topology, the neighborhood notion cannot be defined. This problem can be solved by considering the spatial neighborhood and not the topological one. More specifically, the radius of influence of particles has to be increased from $R1$ to $2 * n * R0$, where n is the ring order (Fig. 9). In fact, in order to avoid getting a voluminous surface, the threshold must also be increased according to the following empirical formula:

$$T = (2 * n - 1) * a$$

This way the isosurface becomes smoother but with loss of surface elements (i.e., loss of particles that do not have neighbors, or specifically particles which their kernel height is less than the threshold T). The lost particles can be retrieved using a second threshold

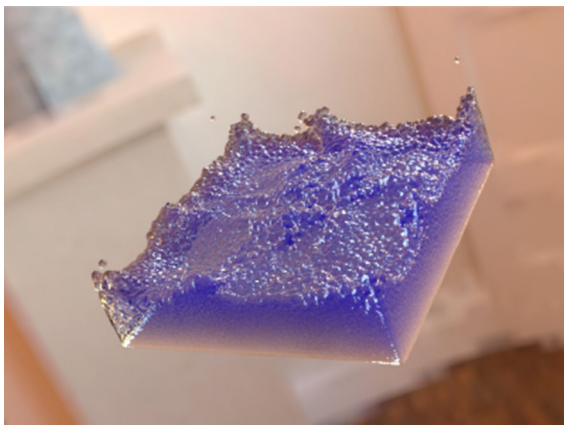

Fig. 7 Resulting surface is slightly smooth on the overlap regions using influence radius R1

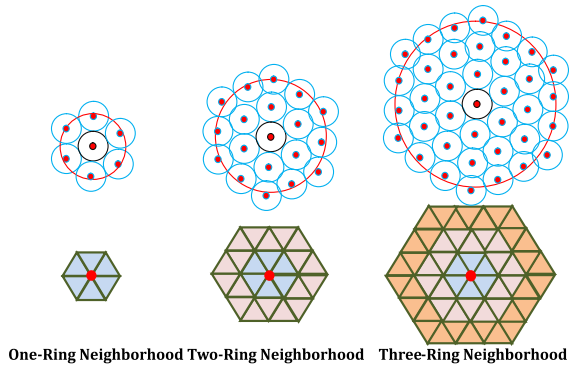

Fig. 8 Notion of topological neighborhood (in a triangular mesh) and spatial one (in a particle cloud)

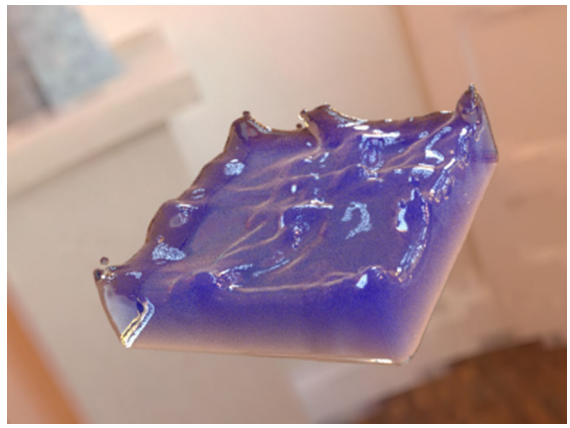

Fig. 9 Resulting surface when $R1 = 2 * 3 * R0$ (3-ring neighborhood) is smoother than the previous one but with loss of some surface elements

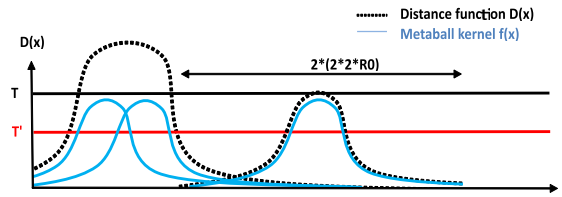

Fig. 10 Distance function of metaballs with a radius resized to $R1 = 2 * 2 * R0$

T' that corresponds to the 1-ring neighborhood (Fig. 10).

$$T' = (2 * 1 - 1) * a = a$$

4.3 Path Tracer Framework

As mentioned previously, particles are defined by two radii, the real radius given by the simulator $R0$, and the influence radius $R1 = 2 * n * R0$.

Particles are organized into two uniform grids with two resolutions. The coarser grid has a cell length of $2 * R1$, while the finer one has a cell length of $2 * R0$.

The framework has two phases:

- *First phase—surface identification* in this phase, particles of the coarser grid are raytraced. First, the ray-sphere intersection points are stored in a 1D-array. Then the surface is determined by sampling the array using the iterative false position method. Once the ray-surface intersection (x) is found, the normal is computed as mentioned above. From this point, secondary rays can be generated (reflection and refraction).
- *Second phase—density integral calculation* for the purpose of generating scattering rays inside the medium, particles of the finer grid are raytraced. First, the intersection points are stored in a 1D-array. Then, Algorithm 1 is applied to calculate the integrated density which corresponds to a position T over the ray. From this point, new scattering rays are generated in random directions according to the phase function.

These two phases must be repeated whenever the ray bounces inside the medium. Ideally, the ray refracts two times; at the front-facing surface and back-facing surface. But in general, when the medium is a liquid (water for example), its refraction index is higher than air. As a result of this, the ray at the back-facing surface might not leave the medium if the incidence angle, denoted (θ_{in}), is greater than a critical angle ($\theta_{critical}$):

$$\theta_{in} > \theta_{critical}$$

This situation is called Total Internal Reflection, and according to Snell's law, the ray should rather be reflected in the medium with the same angle: $\theta_{out} = \theta_{in}$ (see Fig. 11).

In such a case, the ray undergoes several refractions before leaving the medium.

4.4 Extended Path Tracing for Seawater

Using our SPH fluid path tracer, seawater can be handled by specifying how rays should act when a surface formed by different particle types is hit (Fig. 12).

Particles can be either discrete or continuous, and they may have Bidirectional Scattering Distribution Function (BSDF) or BRDF material. Continuous particles contribute together to define an implicit surface, whereas discrete ones are individually rendered as spheres of radius $R0$. In fact, discrete particles can be rendered even if they are located inside the medium (not only on the surface). During the second phase of density integral calculation, when a discrete particle is encountered, the current phase is interrupted and secondary rays are generated.

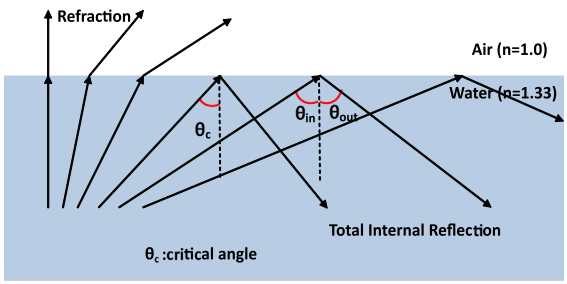

Fig. 11 Ray refraction from a medium with high refraction index to another with low refraction index

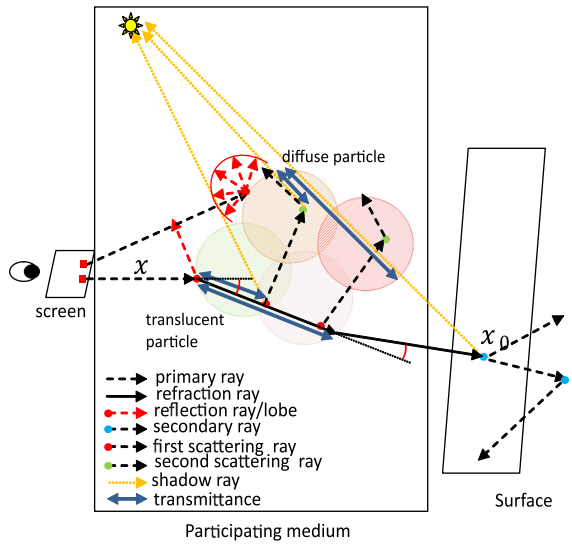

Fig. 12 Path tracing of participating media with refractive boundaries (SPH fluid)

In this paper, the general SPH particle is extended to four particle types, namely: water, foam, bubble, and sand. Unlike previous methods [29–31], bubbles and foam are not rendered as effects but are physically rendered by defining how they should interact with light.

4.4.1 SPH Water

The SPH particle is extended for handling water-type particles by adding the following properties:

- Surface type: continuous.
- Material: BSDF; Dirac delta function with Fresnel IOR equals to 1.33/1.0.
- Albedo: any color.
- Density: unchanged (or 0 for transparent water).

4.4.2 SPH Foam

The SPH particle is extended for handling foam-type particles by adding the following properties:

- Surface type: none.
- Material: none.
- Albedo: white.
- Density: unchanged.

The foam is rendered as a volume because it does not have refractive boundaries and only scattering term is calculated.

4.4.3 SPH Bubble

The SPH particle is extended for handling bubble-type particles by adding the following properties:

- Surface type: discrete.
- Material: BSDF; Dirac delta function with Fresnel IOR equals to 1.0/1.33.
- Albedo: no color.
- Density: 0.

4.4.4 SPH Sand

The SPH particle is extended for handling sand-type particles by adding the following properties:

- Surface type: discrete.
- Material: BRDF; Lambertian diffuse reflectance.
- Albedo: brown.
- Density: infinity.

5 Experimental Results

The experimentation is made on simulation data of a dam break (given to us by the authors of [4]). Our solution (particle-based volumetric path tracing) is implemented on the Mitsuba Renderer [16] and each frame has 768×574 pixels and is computed with 128 samples and 8 bounces on an *Intel* machine (CPU: i7 of 2.60 GHz, RAM:8 GB).

The dam break at a specific time step (which contains 110,375 particles) is rendered by path tracing with our metaball model (described in Sect. 4.2.2) while varying the neighborhood ring from one to three as shown in Fig. 13a–c. Note that the particle density has been set to zero in order to render only the isosurface (i.e., consider only reflection/refraction events and omit scattering ones). It is clear that the surface becomes smoother when the neighborhood ring increases. On the other hand, Houdini (which is a software by Side Effects) has been used to reconstruct a meshed surface from the particle model. Afterward, the obtained surface has been raytraced with the Mitsuba Renderer with the same number of samples and bounces (see Fig. 13d). The resulting image is considered as the reference image with which we compare with the three previous images using the Mean Square Error (MSE) metric as shown in Fig. 15b. We observe that the error decreases as the ring order increases. Beyond the 3-ring, images lose their quality (compared with the reference image) and the minimum MSE that can be obtained is 0.0048 (or 0.48%) which corresponds to the 3-ring neighborhood.

As seen in Fig. 14, we have rendered the particle model from two different viewpoints (using our metaball model with the 3-ring neighborhood). First, we compute only the front-facing surface refraction. Then, both front-facing and back-facing surfaces refractions. Finally, we take into consideration all possible refractions (limited to the number of bounces, namely, 8). Fig. 15a shows the measured MSE between these three images and the reference one. Unlike previous methods [20, 26, 31, 34, 35] which simplify things by computing at most two refractions,

Fig. 13 Dam break rendered: **a** as isosurface with normal estimation from 1-ring neighborhood, **b** as isosurface with normal estimation from 2-ring neighborhood, **c** as isosurface with normal estimation from 3-ring neighborhood, **d** by ray tracing the triangulated mesh of the particle set (reference image)

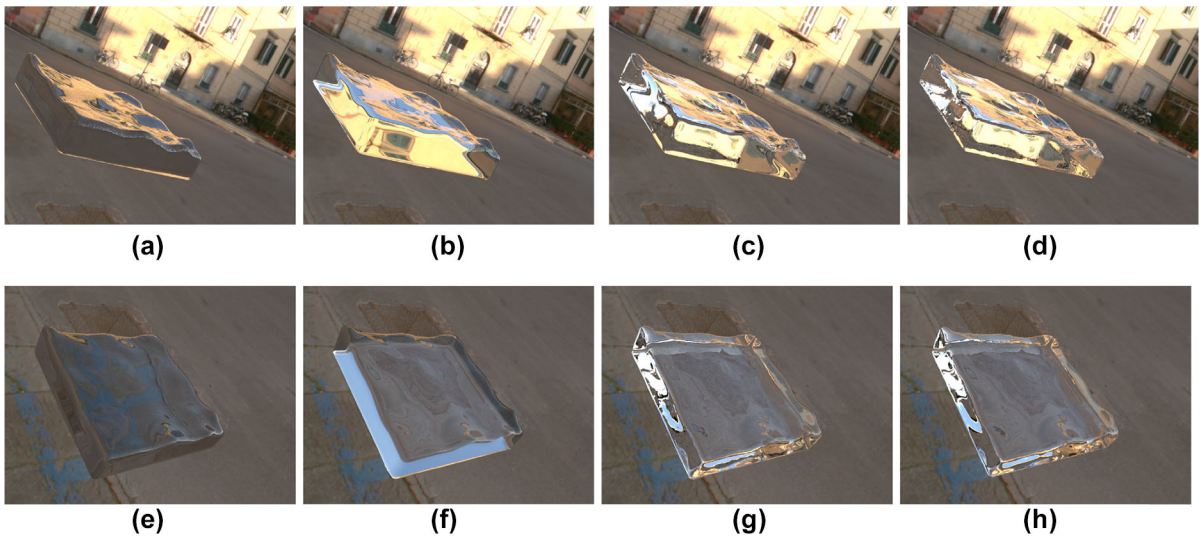

Fig. 14 Dam break rendered from two viewpoints; view 1 (top) and view 2 (bottom): **a**, **e** as isosurface with one refraction, **b**, **f** as isosurface with two refractions, **c**, **g** as isosurface with multiple refractions, **d**, **h** by ray tracing the triangulated mesh of the particle set (reference image)

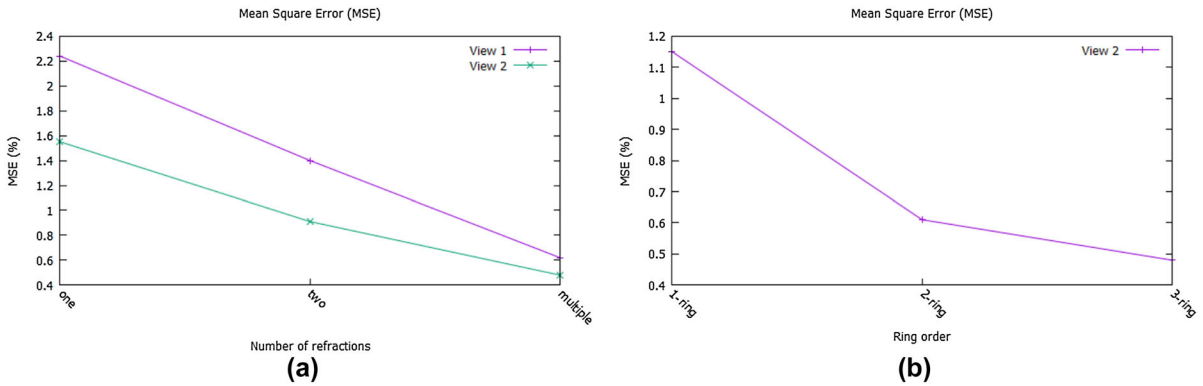

Fig. 15 Mean square error (MSE) between: **a** The images generated with one/two/multiple refractions, and the reference image (see Fig. 14). **b** the images generated using normal estimation from the first/second/third ring neighborhood, and the reference image (see Fig. 13)

Fig. 16 Rendering of the SPH-based fluid as a participating medium with volumetric path tracing (computation of reflection/refraction/scattering events): **a** using our particle-based approach, **b** using the triangulated mesh and the particles transformed into a grid (reference image). MSE = 0.09%

we take into account all possible refractions in order to obtain the closest result to the reference.

As illustrated in Fig. 16, our particle-based approach renders the particles while considering and computing all the events (reflection, multiple refractions, multiple scattering). We note that we created a reference scene by following the scheme shown in Fig. 17. More precisely, First, we transform the particle density into a Eulerian grid, then we render it (along with the triangulated surface already reconstructed by Houdini) by volumetric path tracing on the Mitsuba Renderer. Note that the two results are very similar in quality and the measured MSE is 0.09%.

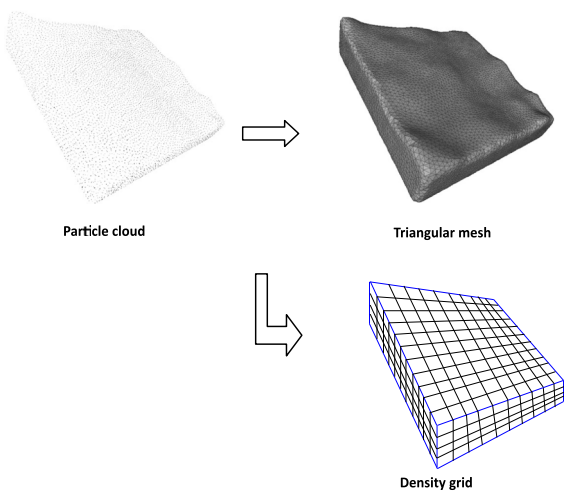

Fig. 17 Reference scene preparation

Nevertheless, our approach needs more time as seen in Table 1

The four particle types (water, foam, bubble, and sand) are shown in Fig. 18. We have also tested our SPH path tracer by rendering simulation data of another solver (Fluids v.3 [27]) that contain 65K particles as depicted in Fig. 19. Actually, Fig. 19a shows transparent water-type particles which have a BSDF material, whereas Fig. 19b demonstrates the existence of two fluids with different colors (red and blue). As regards Fig. 19c, it displays some bubble-type and sand-type particles among numerous transparent water-type particles. Finally, a set of foam-type particles are localized on the top of the fluid in Fig. 19d.

The experimental results, obtained by rendering several simulation data provided by different solvers, clearly demonstrate the efficiency of our approach in terms of (1) implicit surface quality (including normal estimation), and (2) light transport effects (more specifically, multiple refractions and scattering).

The images produced by our approach (see Figs. 14, 16), which considers the physical light effect of multiple refractions, are more realistic than previous approaches which take into account only one or two refractions [20, 26, 31, 34, 35]. Even by considering four refractions, as presented in [15], the resulting images cannot be close to the reference [15]. Another advantage of our fully particle-based approach lies in the fact that bubbles, sand, and foam

Table 1 Time of rendering with all light transport effects

Scene	Ours approach	Meshed surface with density grid
Dam break 'view 2'	7 min 07 s	5.1 s

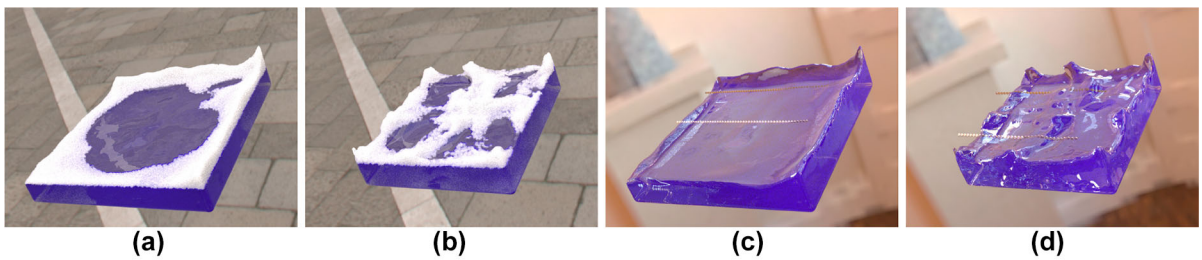

Fig. 18 **a** Foam on stable dam break, **b** foam on turbulent dam break, **c** bubbles and sand in stable dam break, **d** bubbles and sand in turbulent dam break

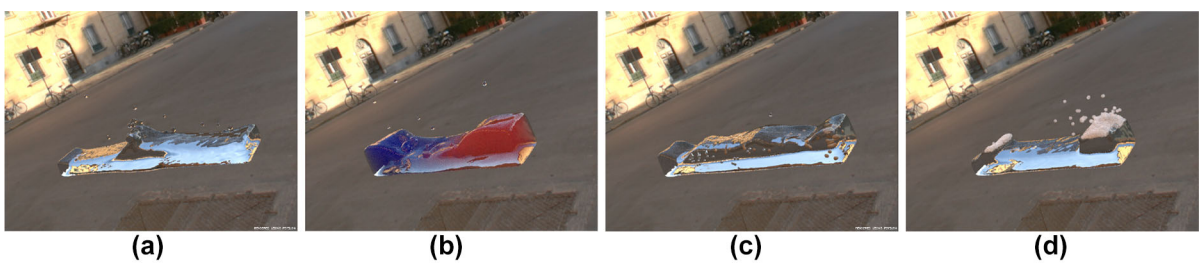

Fig. 19 **a** Continuous particles with BSDF material, **b** two fluids: red and blue, **c** bubbles and sand, **d** foam

are directly visualized by rendering the particles (after some properties changes), without resorting to additional techniques (such as texturing) like previous approaches [29, 31].

Our proposal has the advantage of skipping the intermediate steps of surface extraction and particle transformation, and consequently, the raw-data produced by fluid solvers are directly rendered. However, like any other physically-based rendering technique, the proposed solution needs time because, first, the implicit surface tracking (which includes normal estimation), and second, the light transport effects handling (especially multiple refractions) require extra computation time (as shown in Tables 2, 3).

6 Conclusion

In this paper, we proposed a volumetric path tracing approach for SPH-simulation data. This solution is able to render fluids with high quality. As the evaluation results show, light transport effects (such as reflection, multiple refraction, absorption, and multiple scattering) have been efficiently produced. More precisely, our approach is a fully particle-based approach in which (1) a smooth surface is implicitly tracked, (2) the normal is fixed on the base of the spatial neighborhood, and (3) the ray passing through the medium is sampled according to the integrated density of the particles. Accordingly, the proposed

Table 2 Time of rendering with normal estimation from different neighborhood rings

Scene	Isosurface (1-ring)	Isosurface (2-ring)	Isosurface (3-ring)	Meshed surface
Dam break 'view 2'	2 min 25 s	5 min 02 s	7 min 45 s	1.70 s

Table 3 Time of rendering with different numbers of refractions

Scene	Isosurface (1 refrac.)	Isosurface (2 refrac.)	Isosurface (multi. refrac.)	Meshed surface
Dam break 'view 1'	4 min 52 s	6 min 30 s	8 min 22 s	1.81 s
Dam break 'view 2'	4 min 08 s	6 min 01 s.	7 min 45 s	1.70 s

approach is not only able to render several fluids with different characteristics (color, phase function, etc.), but also several surface types (continuous/discrete) with different materials (BRDF/BSDF). It is worth noting that this work can be integrated into the Mitsuba Renderer as a new module (integrator) in order to directly render SPH simulation data without pre-processing phases (surface reconstruction and particle transformation).

Even though the proposed solution is implemented on the multithreaded Mitsuba Renderer, it can be further accelerated using modern hardware capabilities like CUDA for fast/efficient rendering.

Acknowledgements Special thanks go to our colleague Mohammed A. Merzoug, Assistant Professor at the University of Batna 2, for his help. We are grateful for the time and pertinent comments given by the reviewers. This work was partly supported by the PROFAS grant.

References

1. Adamson, A., & Alexa, M. (2003). Ray tracing point set surfaces. In *Shape modeling international* (pp. 272–279). IEEE.
2. Alexa, M., Behr, J., Cohen-Or, D., Fleishman, S., Levin, D., & Silva, C. T. (2003). Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics*, *9*(1), 3–15.
3. Blinn, J. F. (1982). A generalization of algebraic surface drawing. *ACM Transactions on Graphics (TOG)*, *1*(3), 235–256.
4. Brousset, M., Darles, E., Meneveaux, D., Poulin, P., & Crespin, B. (2016). Simulation and control of breaking waves using an external force model. *Computers & Graphics*, *57*, 102–111.
5. Co, C. S., Hamann, B., & Joy, K. I. (2003). Iso-splatting: A point-based alternative to isosurface visualization. In *Proceedings of 11th Pacific conference on computer graphics and applications, 2003* (pp. 325–334). IEEE.
6. Desbrun, M., & Gascuel, M. -P. (1996). Smoothed particles: A new paradigm for animating highly deformable bodies. In *Computer animation and simulation' 96* (pp. 61–76). Springer.
7. Drebin, R. A., Carpenter, L., & Hanrahan, P. (1988). Volume rendering. In *ACM siggraph computer graphics* (Vol. 22, pp. 65–74). ACM.
8. Enright, D., Fedkiw, R., Ferziger, J., & Mitchell, I. (2002). A hybrid particle level set method for improved interface capturing. *Journal of Computational Physics*, *183*(1), 83–116.
9. Fedkiw, R., Stam, J., & Jensen, H. W. (2001). Visual simulation of smoke. In *Proceedings of the 28th annual conference on computer graphics and interactive techniques* (pp. 15–22). ACM.
10. Gatzke, T. D., & Grimm, C. M. (2006). Estimating curvature on triangular meshes. *International Journal of Shape Modeling*, *12*(01), 1–28.
11. Gingold, R. A., & Monaghan, J. J. (1977). Smoothed particle hydrodynamics: Theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society*, *181*(3), 375–389.
12. Goswami, P., Schlegel, P., Solenthaler, B., & Pajarola, R. (2010). Interactive SPH simulation and rendering on the GPU. In *Proceedings of the 2010 ACM siggraph/eurographics symposium on computer animation* (pp. 55–64). Eurographics Association.
13. Hadwiger, M., Sigg, C., Scharsach, H., Bühler, K., & Gross, M. (2005). Real-time ray-casting and advanced shading of discrete isosurfaces. In *Computer graphics forum* (Vol. 24, pp. 303–312). Wiley Online Library.
14. Holzschuch, N. (2015). Accurate computation of single scattering in participating media with refractive boundaries. In *Computer graphics forum* (Vol. 34, pp. 48–59). Wiley Online Library.
15. Imai, T., Kanamori, Y., & Mitani, J. (2016). Real-time screen-space liquid rendering with complex refractions. *Computer Animation and Virtual Worlds*, *27*(3–4), 425–434.
16. Jakob, W. (2010). Mitsuba renderer.
17. Jarosz, W., Nowrouzezahrai, D., Sadeghi, I., & Jensen, H. W. (2011). A comprehensive theory of volumetric radiance estimation using photon points and beams. *ACM Transactions on Graphics (TOG)*, *30*(1), 5.
18. Jensen, H. W. (2001). *Realistic image synthesis using photon mapping*. Natick: AK Peters Ltd.
19. Jensen, H. W., & Christensen, P. H. (1998). Efficient simulation of light transport in scenes with participating media using photon maps. In *Proceedings of the 25th annual conference on computer graphics and interactive techniques* (pp. 311–320). ACM.
20. Kanamori, Y., Szego, Z., & Nishita, T. (2008). GPU-based fast ray casting for a large number of metaballs. In *Computer graphics forum* (Vol. 27, pp. 351–360). Wiley Online Library.

21. Kulla, C., & Fajardo, M. (2012). Importance sampling techniques for path tracing in participating media. In *Computer graphics forum* (Vol. 31, pp. 1519–1528). Wiley Online Library.
22. Lafortune, E. P., & Willems, Y. D. (1996). Rendering participating media with bidirectional path tracing. In *Rendering techniques' 96* (pp. 91–100). Springer.
23. Lorensen, W. E., & Cline, H. E. (1987). Marching cubes: A high resolution 3D surface construction algorithm. In *ACM siggraph computer graphics* (Vol. 21, pp. 163–169). ACM.
24. Losasso, F., Gibou, F., & Fedkiw, R. (2004). Simulating water and smoke with an octree data structure. In *ACM transactions on graphics (TOG)* (Vol. 23, pp. 457–462). ACM.
25. Monaghan, J. J. (1992). Smoothed particle hydrodynamics. *Annual Review of Astronomy and Astrophysics*, *30*(1), 543–574.
26. Müller, M., Charypar, D., & Gross, M. (2003). Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM siggraph/eurographics symposium on computer animation* (pp. 154–159). Eurographics Association.
27. Rama, C. H. (2012). Fluids v. 3-a large-scale, open source fluid simulator, December 2012, 1(2). http://fluids3.com.
28. Stam, J., & Fiume, E. (1995). Depicting fire and other gaseous phenomena using diffusion processes. In *Proceedings of the 22nd annual conference on computer graphics and interactive techniques* (pp. 129–136). ACM.
29. Takahashi, T., Fujii, H., Kunimatsu, A., Hiwada, K., Saito, T., Tanaka, K., & Ueki, H. (2003). Realistic animation of fluid with splash and foam. In *Computer graphics forum* (Vol. 22, pp. 391–400). Wiley Online Library.
30. Thürey, N., Sadlo, F., Schirm, S., Müller-Fischer, M., & Gross, M. (2007). Real-time simulations of bubbles and foam within a shallow water framework. In *Proceedings of the 2007 ACM siggraph/eurographics symposium on computer animation* (pp. 191–198). Eurographics Association.
31. van der Laan, W. J., Green, S., & Sainz, M. (2009). Screen space fluid rendering with curvature flow. In *Proceedings of the 2009 symposium on interactive 3D graphics and games* (pp. 91–98). ACM.
32. Wald, I., & Havran, V. (2006). On building fast kd-trees for ray tracing, and on doing that in o (N log N). In *IEEE symposium on interactive ray tracing 2006* (pp. 61–69). IEEE.
33. Wang, B., Gascuel, J. -D., & Nolzschuch, N. Point-based light transport for participating media with refractive boundaries. In *Proceedings of the eurographics symposium on rendering: Experimental ideas & implementations, EGSR '16*, Goslar Germany, Germany (pp. 109–119). Eurographics Association. https://doi.org/10.2312/sre.20161216. ISBN 978-3-03868-019-2. 10.2312/sre.20161216.
34. Wyman, C. (2005). An approximate image-space approach for interactive refraction. In *ACM transactions on graphics (TOG)* (Vol. 24, pp. 1050–1053). ACM.
35. Xiao, X., Zhang, S., & Yang, X. (2017). Real-time high-quality surface rendering for large scale particle-based fluids. In *Proceedings of the 21st ACM siggraph symposium on interactive 3D graphics and games* (pp. 12). ACM.
36. Yu, J., & Turk, G. (2013). Reconstructing surfaces of particle-based fluids using anisotropic kernels. *ACM Transactions on Graphics (TOG)*, *32*(1), 5.
37. Zhu, Y., & Bridson, R. (2005) Animating sand as a fluid. In: *ACM Transactions on Graphics (TOG)*, volume 24, pages 965–972. ACM.
38. Zwicker, M., Pfister, H., Van Baar, J., & Gross, M. (2001). Surface splatting. In: *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 371–378. ACM.