Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2007)
D. Metaxas and J. Popovic (Editors)

Animation of Chemically Reactive Fluids Using a Hybrid Simulation Method

Byungkwon Kang Yoojin Jang Insung Ihm

Department of Computer Science, Sogang University, Korea

Abstract

Chemical phenomena abound in the real world, and often comprise indispensable elements of visual effects that are routinely created in the film industry. In this paper, we present a hybrid technique for simulating chemically reactive fluids, based on the theory of chemical kinetics. Our method makes synergistic use of both Eulerian grid-based methods and Lagrangian particle methods to simulate real and hypothetical chemical mechanisms effectively and efficiently. We demonstrate that by modeling chemical reactions using a particle system, an established, physically based fluid system can be extended easily to generate a wide range of chemical phenomena, ranging from catalysis and erosion to fire and explosions, with only a small additional cost.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism – Animation; I.6.8 [Simulation and Modeling]: Type of Simulation – Animation

1. Introduction

1.1. Background and Our Contribution

Chemical phenomena are abundant in the real world, and often comprise important elements of various visual effects that are routinely created in the film industry. Catalysis, erosion, weathering, fire and flame, and explosion are only a few examples of chemical reactions.

When such natural phenomena are to be animated, chemically reactive fluids are one of the most easily applicable ways to depict a wide range of interesting chemical effects. In computational fluid dynamics, reacting fluids have generally been simulated by extending the Navier–Stokes equations to handle the relevant reaction mechanism [Chu02]. However, these general simulation models are often too complicated to be used directly in the generation of animation effects, although a few effective numerical methods have been studied using appropriate assumptions. For example, refer to [Fed97].

In the computer animation community, a simplified computational model was proposed for use in simulating reactive fluids [Gat02], where the behavior of the chemical species in the reaction process was modeled using a transport-reaction system coupled with a simple equation for the energy conservation. More recently, the theory of chemical kinetics was exploited in a simulation process to handle gases containing multiple reacting species [IKC04]. This method, extending the well-accepted fluid simulation model [FSJ01], has been shown to be very effective in generating various reaction phenomena by adopting real or hypothetical chemical mechanisms.

A problem with this simulation scheme [IKC04] is that its entire simulation pipeline is carried out purely on a grid base. In particular, an additional grid buffer must be allocated for each chemical species participating in the reaction mechanism to contain its molar concentration. This often entails an enormous demand for memory space if a complicated reaction mechanism involving many species is to be simulated, or a high resolution grid needs to be adopted for a detailed animation. In addition to the requirement for a large memory space, the additional computation time required for simulating chemical reactions increases cubically with respect to the grid resolution.

In this paper, we present a more effective numerical simulation method for animating chemically reactive fluids. In contrast to previous methods, we exploit a Lagrangian particle system to complement the Eulerian grid-based fluid simulation model. In particular, the computational efficiency is markedly enhanced by computing the chemical reaction step using a simple particle-based computation procedure. It is shown that only a moderate number of particles is required to numerically simulate a chemical reaction process without having to allocate a series of grid buffers holding the molar concentrations of all species.

Our hybrid method is simple to implement, and is flexible

SCA 2007, San Diego, California, August 04 - 05, 2007

enough to enable an animator to create a wide range of visual effects concerning chemical phenomena, ranging from catalysis and erosion to fire and explosions. We demonstrate the effectiveness of our hybrid method by generating several animation examples and analyze their computational performance.

1.2. Related Work

In the graphics community, little effort has been expended toward the general modeling of chemically reactive fluids, except for the results described in Ref. [Gat02, IKC04]. On the other hand, many computer graphics techniques can simulate natural phenomena that are directly or indirectly related to chemical processes. Several researchers have applied the concept of chemical reactions to the generation of natural phenomena, such as stone weathering effects [DEJ*99] and seashell patterns [FMP92]. It has also been shown that chemical mechanisms are effective in the synthesis of textures on arbitrary manifolds [Tur91, WK91].

While not involving the direct application of chemical kinetics, special effects in regard to reactive flows have been simulated frequently in computer animations. Ad hoc or physically motivated methods have been explored to simulate fire and flame in numerous studies [CMTM94, SF95, BPP01, NFJ02, LF02, WLMK02, LSF06]. Animating realistic explosions has also been a big challenge in computer animation, and diverse practical simulation schemes have been presented [NF99, MMA99, YOH00, BY01, FOA03, RNGF03, SRF05]. Recently, surface reactions between multiple interacting liquids were considered in a multiphase level set framework [LSSF06].

While grid-based methods prevail in physically based fluid animation, particle-based methods have also been explored. A set of particles were used to solve diffusion type equations that modeled gaseous phenomena, overcoming the memory overhead caused by grid-based schemes [SF95]. Since it was applied to the animation of viscous fluids [DG96], the Smoothed Particle Hydrodynamics method and other hybridized variants have been exploited in liquid animation [MCG03, MSKG05, PTB*03, ZB05, CBP05].

Lagrangian particles have often been explored as a means of complementing the Eulerian grid-based schemes. As well as being utilized to enhance the front-tracking process [FF01, EMF02, LSSF06], particles have been directly applied to the creation of fine details of liquids, such as droplets, splashes, and bubbles that are difficult to create with the purely grid-based solutions [Sim90, OH95, FF01, TFK*03, GH04, GSLF05, KCC*06].

In addition to liquids, particles have also been used in simulations of gaseous phenomena. A particle system was combined with a grid-based simulation method to provide a stable and effective animation of explosions, where suspended particles advected in the grid space were used to model the motion of particulate fuel and combustion products [FOA03]. Particles were also proven to be effective in generating large-scale explosions [RNGF03], while a purely particle-based method that employs flame and air particles was presented for the simulation of explosive flames [TOT*03]. Vorticity-carrying particles have also been utilized to create the turbulent appearance of fluids in both Eulerian and Lagrangian simulation schemes [AN05, PK05, SRF05].

2. Particles and Grids in Simulations

Our hybrid simulation scheme couples Lagrangian particle-based methods with Eulerian grid-based methods to effectively exploit the advantages of both schemes.

2.1. Grids

In our method, Eulerian grids were used to model fluids that carried chemical species, while Lagrangian particles were employed primarily to simulate reaction mechanisms. On these grids, only attributes such as the velocity, the density, the temperature, and the external force were defined. This contrasts with previous simulation methods [IKC04] that need to allocate as many grid buffers as the number of participating species. As will be shown later, separating the reaction computation from the entire simulation pipeline avoids an impractical demand on the memory required, which is often needed when dealing with a complicated reaction mechanism, or when a high-resolution simulation grid is desired. In addition, it speeds up the simulation process.

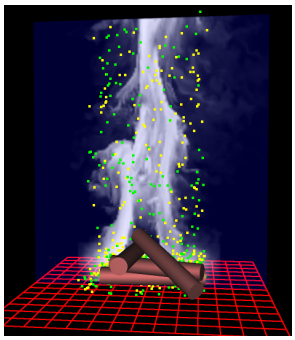
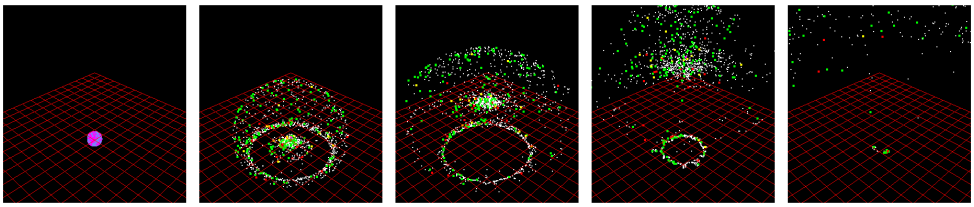
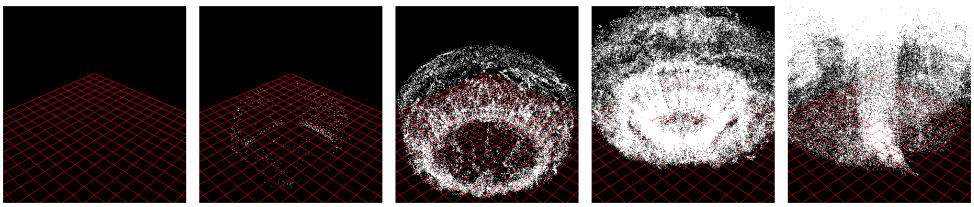


Figure 1: *Hybrid representation of a reaction mechanism, $S_1 + S_2 \xrightarrow{r_1} 2S_3$. This usually produces a very effective result when the species in the reaction mechanism are assigned different duties. In this example hypothetical reaction, applied for the purpose of generating the campfire scene shown in Figure 5(a), the reactants, S_1 and S_2, denoted by the green and yellow particles, respectively, have the role of triggering the reaction process. On the other hand, the resulting product, S_3, in volumetric density aims to represent the fluid to be visualized in the rendering stage.*

Our simulation scheme is different from another related method [FOA03] in that a hybrid representation of fluids (and chemical species) was possible. In the previous method, the fluid itself was modeled using particles, and basically rendered using a particle-rendering method. Our



(a) Material and vortex particles



(b) Soot particles

Figure 2: *Particles used in the simulation. These images demonstrate how the three types of particles move around the space when the example scene shown in Figure 5(b) was simulated. In (a), the colored dots (i.e., the larger dots) represent material particles, while the white dots (i.e., the smaller dots) model vortex particles. On the other hand, soot particles are illustrated in (b). The material particles led the simulation process as the reaction mechanism, computed numerically using them, basically determined the overall behavior of the reactive fluids.*

method also allowed us to model the fluid to be rendered using particles, and provided another option where the chemical species could be partitioned into two classes, depending on their roles in the simulation. The *Lagrangian species* were those that were represented by particles throughout the simulation. Their function was to bring about a chemical reaction that affected the overall flow of the fluid. On the other hand, the *Eulerian species* were those species whose role was to model the fluid to be visualized in the rendering stage. Unlike the Lagrangian species, these are represented using a volumetric density field in the grid space. When chemical species of this class were produced as a consequence of the reaction, the increase in their molar concentration was converted to the equivalent amount in density, and was reflected in the density field. By adopting this hybrid scheme, we could exploit state-of-the-art volume-rendering techniques, creating fine-scale detail of simulated fluids while limiting the number of particles used in the simulation to a moderate size. See Figure 1.

2.2. Material, Soot and Vortex Particles

During a simulation, up to three types of particle can be involved (refer to Figure 2). First, *material particles*, which are mandatory, model the chemical species participating in the chemical reaction system. Each of these represents a single species, and is associated with a descriptor that consists of its position (3), velocity (3), and radius of influence (1), as well as its species indicator, S_i (1), and molar concentration, $[S_i]$ (1). The figure in parentheses indicates the number of data fields necessary for representing the corresponding attribute, where a one-byte unsigned character is adequate for S_i, while floating-point numbers must be used for the others.

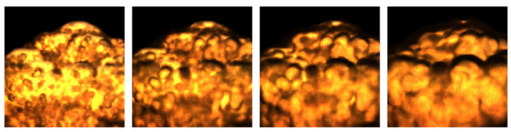


Figure 3: *Soot particles in rendering. In the rendering stage, the soot particles, if used, are converted to volumetric density data using a Gaussian smoothing kernel, which is then used to depict solid combustion products. Depending on their radii of influence, a wide range of appearances can be created in rendering.*

When the initial locations and molar concentrations of all the chemical species are described by an animator, the number of material particles to be placed in the reaction system is determined per chemical species, proportional to the given concentration. Then, these new particles are distributed in the specified areas in a jittered fashion. Once added, they react with the other reactants already in the reaction system. To ease the computational burden in handling the particle set, material particles whose concentrations fall below a threshold level during the reaction process are treated as being insignificant, and are eliminated immediately from the particle system.

The second class of particles, *soot particles*, have been shown to improve the rendering quality by depicting solid combustion products in a realistic manner [FOA03]. These

are created, if needed, in each cell in the grid space whose temperature falls below a threshold temperature level (see Subsection 3.4.4 for details on how the soot particles are generated). Initially, each soot particle is assigned a short life span that decreases with increasing time. The soot particles diminish from the reaction system when their span expires. These are massless, and do not affect the reaction process. Each soot particle has a description that includes its position (3), remaining life span (1), and radius (1) that is used to decide its extent in the rendering stage. Notice that the soot particles are optional and ignored when an applied chemical reaction does not create any soot.

We also adopted optional *vortex particles* to create the detailed turbulent appearance of a fluid. As utilized successfully in previous methods, for example, [SRF05], these particles are intended to reintroduce small-scale details lost when the fluid is simulated using a semi-Lagrangian scheme. In our method, they are generated in each cell of the grid proportional to the intensity of the chemical reaction that occurred in the cell. The vortex particles can then affect the velocity field via the vorticity confinement force, as will be explained later. The description of a vortex particle consists of its position (3), direction (3), and radius (1).

3. Simulation Methods

For a consistent explanation of our method, we adopted the following hypothetical chain mechanism that involved six chemical species.

$$\left.\begin{array}{l} S_1 \xrightarrow{r_1} 2S_4, \quad S_2 \xrightarrow{r_2} 2S_5, \quad S_4 + S_2 \xrightarrow{r_3} S_6 + S_5 \\ S_5 + S_1 \xrightarrow{r_4} S_6 + S_4, \quad S_1 + S_2 + S_6 \xrightarrow{r_5} S_6 + 5S_3 \end{array}\right\} \quad (1)$$

This artificial chain reaction mechanism, similar to a gas-phase hydrogen–oxygen explosion, is well suited to the modeling of very rapid reactions, such as explosions. Here, it is assumed that two highly combustible materials, S_1 and S_2, react instantaneously with each other with the help of intermediate materials, S_4, S_5, and S_6, to produce the remnant of reaction S_3, where the exponentially growing radicals, S_4 and S_5 trigger a very fast reaction (refer to Ref. [Lev02] for more detail). In this example mechanism, S_3 works as a Eulerian species, while the others are regarded as Lagrangian species.

3.1. Update of Fluid Velocity (Step 1)

The velocity, $\mathbf{u}$, of the fluid is updated in the grid space using the Euler equation. $\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u}\cdot\nabla)\mathbf{u} - \frac{1}{\rho}\nabla p + \frac{\mathbf{f}}{\rho}$, where p, ρ and $\mathbf{f}$ denote the fluid pressure, the density, and the external force acting on the fluid, respectively. When an explosive fluid is to be simulated, it is important to be able to properly control the abrupt expansion or contraction of reacting fluids. As introduced in previous work [FOA03], and used later by an ensuing method [IKC04], we applied the modified divergence equation, $\nabla\cdot\mathbf{u} = \phi$, where the constraint, ϕ, controls the expansion or contraction of the fluid due to the chemical reaction. In our method, this is adjusted at each voxel in the final step of the simulation, reflecting the reaction that takes place in the corresponding cell region.

3.2. Advection of Fluid Density and Temperature (Step 2)

Once the velocity field is computed, the density, d, and the temperature, T, of the fluid are evolved through the updated velocity field using the advection–diffusion equations, $\frac{\partial d}{\partial t} = -(\mathbf{u}\cdot\nabla)d + \kappa_d\nabla^2 d$ and $\frac{\partial T}{\partial t} = -(\mathbf{u}\cdot\nabla)T + \kappa_T\nabla^2 T + H_T$, with diffusion coefficients, κ_d and κ_T. In the second equation, H_T denotes a heat source term that represents the increase or decrease in heat energy induced by the chemical reaction.

3.3. Advection of Particles (Step 3)

Once the first two Eulerian steps are complete, the particles in the reaction system are moved through the fluid. First, the massless soot and vortex particles are advected through the velocity field, $\mathbf{u}$, using a simple rule, $\mathbf{x}_P^{t+\Delta t} = \mathbf{x}_P^t + \mathbf{u}_{\mathbf{x}_P}^t \cdot \Delta t$, where the position, $\mathbf{x}_P$, of a particle, P, is modified using the trilinearly interpolated velocity, $\mathbf{u}_{\mathbf{x}_P}$.

The material particles, on the other hand, are treated differently, as they convey mass as well as their own velocity, $\mathbf{v}_P$. Once they are moved using $\mathbf{v}_P$, as $\mathbf{x}_P^{t+\Delta t} = \mathbf{x}_P^t + \mathbf{v}_P^t \cdot \Delta t$, their velocity is updated using the equation $\mathbf{v}_P^{t+\Delta t} = \mathbf{v}_P^t + \frac{\mathbf{f}_{\mathbf{x}_P}}{m_P} \cdot \Delta t$. Here, m_P denotes the mass of the particle, P, that can be expressed as the product of its molar concentration, $[S_i]$, and molar mass, M_{S_i}. The term, $\mathbf{f}_{\mathbf{x}_P}$, is another physical quantity that must be set properly as a driving force. The first choice for this parameter is the external force $\mathbf{f}$ that is trilinearly interpolated at $\mathbf{x}_P$. An alternative would be to use a function of the fluid's velocity that allows for easier control to introduce some interesting physical phenomena, for example, a drag force. In our implementation, we used a heuristic linear combination of the trilinearly interpolated fluid's velocity and external force for this driving force.

3.4. Application of Chemical Kinetics (Step 4)

The chemical reaction is now ready to take place. The key to an effective simulation of chemical kinetics is to discretize the reaction mechanism numerically. Unlike the method in Ref. [IKC04] that simulates a reaction at each voxel in the grid space, our method utilizes material particles to approximate the kinetics process.

3.4.1. A Quick Review on Chemical Kinetics

Chemical kinetics is a branch of kinetics that studies the mechanisms of chemical reactions, and describes the nonequilibrium states of fluid systems containing several reactive substances. Given the chemical reaction, $aA + bB \longrightarrow eE + fF$, where A, B, E, and F denote the chemical species, and a, b, e, and f are their stoichiometric coefficients, respectively, the reaction is described by the following differential equations:

$$r = -\frac{1}{a}\frac{d[A]}{dt} = -\frac{1}{b}\frac{d[B]}{dt} = \frac{1}{e}\frac{d[E]}{dt} = \frac{1}{f}\frac{d[F]}{dt},$$

where the rate of reaction, r, determines the change in molar concentration, $[\cdot]$ (in mol/L) of the reactants A and B, and the products E and F.

The rate of reaction is a time-dependent function, $r = f_r([A],[B],[E],[F],t)$ of the concentration of species present at a time, t. For a large class of chemical reactions, it is found experimentally to be proportional to the concentration of each reactant and/or product raised to a given power. For example, if only a forward reaction occurs, then it can be expressed in the form, $r = f_r([A],[B],t) = k[A]^{\alpha}[B]^{\beta}$, for orders α and β, where the rate constant, k, is a function of temperature and pressure. Usually, the constant depends strongly on the temperature, while the pressure dependence is usually ignored. From the viewpoint of fluid animation, the rate function may be treated as a control parameter that is determined by an animator. For more details on the theory of chemical kinetics, refer to a textbook on physical chemistry, for example, in Ref. [Lev02].

Recall the multistep mechanism given in Eq. (1), in which r_e ($e = 1,2,\cdots,5$) denotes the rate of the eth reaction. Assuming that only a forward reaction takes place, then this chemical process is simulated by updating the molar concentrations of the six species in each time frame through a system of ordinary differential equations

$$\begin{aligned}
\frac{d[S_1]}{dt} &= -k_1[S_1]^{\sigma_1} - k_4[S_1]^{\sigma_1}[S_5]^{\sigma_5} \\
&\quad - k_5[S_1]^{\sigma_1}[S_2]^{\sigma_2}[S_6]^{\sigma_6} \\
\frac{d[S_2]}{dt} &= -k_2[S_2]^{\sigma_2} - k_3[S_2]^{\sigma_2}[S_4]^{\sigma_4} \\
&\quad - k_5[S_1]^{\sigma_1}[S_2]^{\sigma_2}[S_6]^{\sigma_6} \\
\frac{d[S_3]}{dt} &= 5k_5[S_1]^{\sigma_1}[S_2]^{\sigma_2}[S_6]^{\sigma_6} \\
\frac{d[S_4]}{dt} &= 2k_1[S_1]^{\sigma_1} - k_3[S_2]^{\sigma_2}[S_4]^{\sigma_4} + k_4[S_1]^{\sigma_1}[S_5]^{\sigma_5} \\
\frac{d[S_5]}{dt} &= 2k_2[S_2]^{\sigma_2} + k_3[S_2]^{\sigma_2}[S_4]^{\sigma_4} - k_4[S_1]^{\sigma_1}[S_5]^{\sigma_5} \\
\frac{d[S_6]}{dt} &= k_3[S_2]^{\sigma_2}[S_4]^{\sigma_4} + k_4[S_1]^{\sigma_1}[S_5]^{\sigma_5}.
\end{aligned}$$

3.4.2. Update of Molar Concentration

The first step to be taken in the chemical reaction step is to update the molar concentrations of the reacting species according to the applied reaction mechanism. This process proceeds by solving the corresponding system of ordinary differential equations numerically. The second-order (or third-order) Runge–Kutta method is usually sufficient for the time discretization. Spatially, on the other hand, a different numerical method must be applied, as the continuous molar concentration field has been discretized by material particles where particulate species of different types are intermingled.

To simulate the chemical process, we traversed the material particles in a random order while integrating the differential equations in the particle space. When a particle, P, at $\mathbf{x}_P$ is visited, then the particle system, organized in a dynamic kd-tree, is searched to locate neighboring material particles that exist in the spherical region defined by the center, $\mathbf{x}_P$, and radius, r_{sp}.

Suppose that n_p particles are found in the neighborhood. They can be viewed as locally approximate the molar concentration field around $\mathbf{x}_P$. To model the local reaction process, we first estimate the concentrations of all the reacting species at $\mathbf{x}_P$. An effective way of doing this is to collect the molar concentration from the neighboring material particles on a species-by-species basis

$$[S_s]_{\mathbf{x}_P} = \sum_{i=1}^{n_p} \delta_{s<P_i>} \cdot w_i(\mathbf{x}_P) \cdot [S_{<P_i>}]_{\mathbf{x}_{P_i}},\ 1 \le s \le 6,$$

where $<P_i>$ denotes the index of the species of the ith-found particle, P_i, and $\delta_{s<P_i>} = 1$ if $s = <P_i>$, or otherwise, zero. In this process, we applied a normalized Gaussian kernel, $w_i(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{3}{2}} r_{sp}^3} e^{-\|\mathbf{x}-\mathbf{x}_i\|^2/2r_{sp}^2}$, to adjust the contribution from each local particle appropriately.

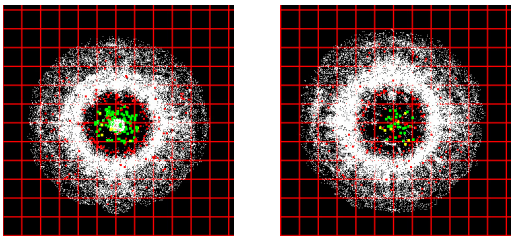
Once collected for all species, these are used to integrate the differential equation system for the time interval, Δt. As a result, we obtain the rate of reactions at $\mathbf{x}_P$ and more importantly the increment, $\Delta[S_s]_{\mathbf{x}_P}$, that indicates the changes in the reactive species resulting from their local chemical reaction. To reflect the reaction phenomena to the particle system, we return the changes to the local material particles by evaluating the following equation for all local particles, $1 \le i \le n_p$, and all species, $1 \le s \le 6$,

$$\begin{aligned}
[S_{<P_i>}]^{t+\Delta t} &= [S_{<P_i>}]^t \\
&\quad + \frac{\delta_{s<P_i>} \cdot w_i(\mathbf{x}_P)}{\sum_{j=1}^{n_p} \delta_{s<P_j>} \cdot w_j(\mathbf{x}_P)} \cdot \Delta[S_s]_{\mathbf{x}_P}.
\end{aligned}$$

During this return process, it may be possible that there are no, or very few, particles of a specific type in the local region that can increase in concentration. In this case, more particles are randomly created in the region so that they can carry a moderate amount of material, which avoids the creation of over-heavy material particles.

Our proposed method solves the system of differential equations on a particle basis through a sequential traversal of the material particles. We must make sure that this sequential visit does not add a serious bias to the solutions, because the reacting particles are interrelated, requiring the system to be integrated simultaneously. A simple and efficient method to prevent a possible bias is to subdivide the time interval, Δt, into n_s segments, and repeat the traversal process for the time interval, $\frac{\Delta t}{n_s}$, each time with a different random order.

Figure 4 shows simulation results generated using n_s = 1 and 5, for the explosion scene shown in Figure 5(b). The scene was designed to create a symmetrical shaped flame, although the randomized placement of material particles caused some asymmetry. Compared to the cases when a multistep integration was carried out (for instance, see figure (b)), a small bias was observed when no subdivision of the time intervals was carried out (figure (a)). However, the difference was very small, and was usually difficult to find even when a ten-step integration method was tested. Considering



(a) 1-step integration (b) 5-step integration

Figure 4: *Particle-based simulation of a chemical reaction. The randomized traversal method offers a reliable space discretization scheme for particle-based integration of the differential equation system. The images contrast the particles produced when a five-step integration technique was applied (seen from the above, just after the explosion (the 10th frame) in the scene shown in Figure 5(b)). Even using a one-step integration, as illustrated in (a), the result shows only a slight bias for such a stiff reaction mechanism. In fact, the difference is quite small, and is usually difficult to find in animation results.*

that the tested scene involved a very stiff reaction mechanism, that is, an explosion, the bias introduced by the proposed sequential visit was small, even when no subdivision was applied, and was often unnoticeable in the rendering images.

Note that, when the scene was simulated using a one-step integration process, the chemical kinetics step (Step 4) took 0.52 seconds per frame on average, as shown in Table 1 in Section 4. Hence, the extra time for the multistep computation, if needed, had only a slight effect on the entire simulation time. In conclusion, the randomized traversal method, coupled with the Gaussian smoothing kernel, provides a reliable integration scheme for the simulation of chemical reactions.

3.4.3. Adjustment of Density and Temperature

In addition to the molar concentration, the chemical reaction process may affect other physical attributes of the reacting fluid. For example, when a Eulerian species is adopted in the simulation, its density must be updated too. In the example reaction mechanism where S_3 is the Eulerian species, $\Delta[S_3]_{\mathbf{x}_P}$, in molar concentration indicates how much has been created as a result of the reaction. In our method, we convert the added material with mass $M_{S_3} \cdot \Delta[S_3]_{\mathbf{x}_P}$ to the equivalent density, and distribute it to the nearby cells contained in the local spherical region. Considering the fact that density is mass divided by volume, we can update the density values at the neighboring voxel, (i, j, k), as:

$$d_{ijk}^{t+\Delta t} = d_{ijk}^{t} + \frac{\frac{w_{ijk}(\mathbf{x}_P)}{w_{total}} \cdot M_{S_3} \cdot \Delta[S_3]_{\mathbf{x}_P}}{V},$$

where V is the volume of cell in the grid space. As before, the distribution of material is weighted by the Gaussian kernel, in which $w_{ijk}(\mathbf{x}_P)$ represents the weight to voxel (i, j, k) with respect to the center of the sphere, and w_{total} is the total sum of all the weights involved. This update process is repeated for all Eulerian species if there are more than one present.

Furthermore, the voxel temperature, T_{ijk}, was also updated, if necessary, through the heat source term H_T in the temperature equation, as explained in the second step. In the all presented animations, we employed a linear control function, $H_T = f_r(r_1, r_2, \cdots, r_5)$ with respect to the reaction rates to define heat generated at the reacting particles, while a different type of control function may be used. The amount of heat produced at each material particle was then distributed to the surrounding voxels as $T_{ijk}^{t+\Delta t} = T_{ijk}^{t} + w_{ijk}(\mathbf{x}_P) \cdot (H_T)_{\mathbf{x}_P}$. By allowing such a user-defined function, the animator can control the flow of reacting fluid through the temperature-buoyancy force-velocity chain.

3.4.4. Generation of Soot and Vortex Particles

For a chemical reaction that generates soot, our simulation scheme enables us to handle soot particles. In previous research [FOA03], these were created when the soot mass had accumulated from sufficient fuel particles. In our method, they are produced when the temperature at a voxel falls below a threshold value after the temperature update process has been carried out. For this, the amount of density to be transformed into soot is first determined proportional to the magnitude of the decrease in temperature. Then, a proper number of soot particles is created and scattered randomly in the corresponding cell region. In addition to its position, each soot particle is assigned two attributes: a lifespan and a radius that decrease slowly with increasing time. These attributes are exploited in the rendering stage to represent soot that naturally fades away.

As well as the soot particles, vortex particles are also generated at this stage, with the aim of creating highly turbulent flows in the region of an intense reaction. In particular, to reflect a chemical reaction that actually takes place, we utilized the rates of reaction obtained at the material particles to control the magnitude of the vorticity. To create a vortex particle, the number of new particles, n_{vp}, was determined using a user-controllable function, $n_{vp} = f_{vort}(r_1, r_2, \cdots, r_5)$, again, linear to the reaction rates in our examples, for each material particle with a high rate of reaction. Then, this number of vortex particles was scattered in the spherical region around the material particles with an initial vorticity vector, ω_{vp}, that was based on the gradient field of the rates of reaction. In the case of our example mechanism, we used r_5 as the rate of reaction that produces the Eulerian species, S_3, so that $\omega_{vp} = c_{vort} \cdot \frac{\nabla r_5}{||\nabla r_5||}$, where c_{vort} is a parameter that controls the overall strength of vorticity. After generated, the soot and vortex particles move through the velocity field as massless particles.

Once all these computations were complete, the molar concentration and lifespan attributes of the material and soot particles, respectively, were examined, deleting those that showed insignificant differences from the particle system.

3.5. Update of External Force and Divergence Constraint (Step 5)

A chemical reaction occurring in the simulation domain altered the flow of a fluid. In our model, the reaction process at a material particle modified the external force and the divergence constraint through the rate of reaction that, in turn, affected the computation of the velocity field in the ensuing time frame. The external force, $\mathbf{f} = \mathbf{f}_{buoy} + \mathbf{f}_{vort}$, was defined as the sum of the buoyancy force, $\mathbf{f}_{buoy}$, and the vorticity force, $\mathbf{f}_{vort}$, possibly using another user-controlled force term [FM97, FSJ01, IKC04]. The new density, d, and temperature, T, updated in the previous stages, were reflected in the calculation of $\mathbf{f}_{buoy}$, as $\mathbf{f}_{buoy} = -\alpha d\mathbf{z} + \beta(T - T_{amb})\mathbf{z}$, where $\mathbf{z}$ and T_{amb} are the direction opposite to the force of gravity and the ambient temperature, respectively.

The vorticity force, $\mathbf{f}_{vort}$, was also updated at each voxel from the set of vortex particles. For each vortex particle, the Gaussian kernel was applied in a similar manner as before to accumulate the smoothed vorticity force onto its surrounding voxels. Finally, the divergence constraint, ϕ, was set to adjust the method of expansion or contraction of the reacting gaseous fluid using yet another linear control function, $\phi = f_\phi(r_1, r_2, \cdots, r_5)$, of the rates of reaction.

4. Experimental Results

To show its effectiveness, we implemented our method and tested it using several example scenarios. Figure 5 shows some animations generated using the presented hybrid method and visualized by a photon-mapping based ray tracer, capable of rendering both density and particle data.

Table 1 summarizes the statistics collected on a desktop PC with a 3.2 GHz Intel Pentium 4 CPU and 2 GB RAM for the three test scenes illustrated in Figure 5(a) to (c). The first scene shows a campfire scene generated using very simple chemical kinetics, $S_1 + S_2 \xrightarrow{r_1} 2S_3$, where the fuel was continuously fed through two Lagrangian species, S_1 and S_2, and the flame was represented by a Eulerian species, S_3. On the other hand, the other two explosion scenes were created using the chain reaction mechanism given in Eq. (1), in which the Lagrangian chemical explosives, S_1 and S_2, reacted with each other to explode instantly, producing the Eulerian smoke, S_3.

As can be seen in the step-by-step dissection of the simulation times, the additional cost ('Step3' and 'Step4'), required to extend the established fluid simulation model [FSJ01] to include the chemical kinetics, was very low. Mostly, the first two grid-based steps ('Step1' and 'Step2') for solving the Navier–Stokes equations and advecting the density and temperature dominated the computation time (refer to Table 2 to see how the timings for these two steps depended on the grid resolution). This table also shows that chemical reaction phenomena may be simulated effectively using a moderate number of particles. Only a couple of thousand material particles were needed to trigger the hypothetical chemical reaction to create a natural flame in the campfire scene (Table 1(a)). For the two explosion scenes, the average number of material particles was much less than the maximum number, because most of these burnt out in an instant in the first few frames after ignition (Table 1(b) and (c)).

As implied by the test data, when soot particles were used they usually dominated the particle use. Our method was able to adjust the size of the soot particle set. In the first explosion scene (Table 1(b)), we intended to produce a large enough number of soot particles to depict a detailed and somewhat grainy soot appearance. On the other hand, we controlled the soot parameter to limit the generation of soot particles in the next scene (Table 1(c)) in an attempt to obtain a different rendering appearance. Although we may not be able to directly compare the simulation statistics from previous methods, for example, [FOA03], it is clear that our method utilizes particles quite efficiently in the simulation of chemically reacting fluids.

Table 2 compares our hybrid technique with a previous grid-based method [IKC04]. The scenario used to produce the explosion scene in Figure 5(b), was tested using four different grid resolutions. As revealed in the first table (Table 2(a)), the previous method ('Grid-only method') consumed about twice the memory of our method ('Hybrid method') for the example chain mechanism that employed six chemical species. This is because the former method needs to allocate six additional copies of the grid buffer to hold the respective molar concentrations. On the other hand, our hybrid technique replaces this memory burden by a moderate number of particles (refer to the table in Table 2(b) to see how the numbers of employed particles vary as the resolution increases).

From a temporal aspect, our method also markedly outperformed the previous method that relies on grids only, as revealed in Table 2(c). Recall that the chemical reaction was applied in the grid-based third step ('Step3') in the previous method, while the same task was carried out in the particle-based third and fourth steps ('Step3' and 'Step4') in our method, as explained in Subsection 3.3 and 3.4. Obviously, the differences in timing mainly arose from the chemical reaction steps. As indicated by the table, the particle-based simulation was less dependent on the grid resolution, which allows for easier simulations on higher-resolution grids.

Finally, Figure 5(d) to (g) illustrate an extra set of animation scenes created using the presented method. Scene (d) used seven chemical species in total in such a way that the three different types of gases exploded only when the blue gas met the other gases. On the other hand, scene (e) demonstrates an example where 11 species were intermingled. The last two scenes in figure (f) and (g) show examples where catalytic agents scattered on the surfaces reacted with other reactants to create the interesting animations.

5. Conclusion

In summary, we have presented a hybrid simulation method that is suitable for modeling chemically reactive fluids based on the theory of chemical kinetics. It was built to take advantage of both Eulerian and Lagrangian frameworks. By

(a) Campfire scene (grid resolution: $160 \times 160 \times 180$)

	Simulation time per frame (seconds)					Number of particles per frame		
	Step1	Step2	**Step3**	**Step4**	Step5	Material	Soot	Vortex
Mean	145.01	136.26	0.03	1.35	1.93	2,026	0	55
Max	145.86	137.20	0.05	1.58	2.06	2,378	0	94

(b) Explosion scene 1 (grid resolution: $160 \times 160 \times 200$)

	Simulation time per frame (seconds)					Number of particles per frame		
	Step1	Step2	**Step3**	**Step4**	Step5	Material	Soot	Vortex
Mean	163.66	151.64	4.45	0.52	3.31	119	176,594	584
Max	164.27	152.23	10.08	13.02	4.00	3,999	400,162	886

(c) Explosion scene 2 (grid resolution: $220 \times 200 \times 160$)

	Simulation time per frame (seconds)					Number of particles per frame		
	Step1	Step2	**Step3**	**Step4**	Step5	Material	Soot	Vortex
Mean	224.29	207.17	0.22	0.44	10.96	116	7,487	636
Max	225.75	208.48	0.50	6.31	14.84	2,998	17,887	717

Table 1: *Simulation Statistics. The running times and particles used in our method were measured for three scenarios illustrated in the first three images of Figure 5. The statistics, given as both average and worst case numbers, reveals that the third and fourth steps ('Step3' and 'Step4'), added to extend the established fluid simulation method [FSJ01] for handling reactive fluids, comprised only a small portion of the entire simulation time. Furthermore, the test results also imply that only a moderate number of particles, in particular, material particles, is enough to simulate the chemical reaction mechanism in a natural manner.*

adding a particle-based simulator to a well-accepted grid-based Navier–Stokes equation solver, we effectively animated a wide range of chemical phenomena, including catalysis, erosion, fire and flame, and explosions. From tests on various example scenarios, we found that real or hypothetical chemical reaction mechanisms allowed us to generate natural and interesting animation effects easily employing user-defined control parameters.

Acknowledgements. This research was supported by the Ministry of Information and Communication of Korea under the IT Research Center support program.

References

[AN05] ANGELIDIS A., NEYRET F.: Simulation of smoke based on vortex filament primitives. In *Proc. of ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2005), pp. 87–96.

[BPP01] BEAUDOIN P., PAQUET S., POULIN P.: Realistic and controllable fire simulation. In *Proc. of Graphics Interface 2001* (2001), pp. 159–166.

[BY01] BASHFORTH B., YANG Y.-H.: Physics-based explosion modeling. *Graphical Models 63*, 1 (2001), 21–44.

[CBP05] CLAVET S., BEAUDOIN P., POULIN P.: Particle-based viscoelastic fluid simulation. In *Proc. of ACM SIGGRAPH/Eurographics on Computer Animation 2005* (2005), pp. 219–228.

[Chu02] CHUNG T.: *Computational Fluid Dynamics*. Cambridge University Press, ISBN 0-521-59416-2, 2002.

[CMTM94] CHIBA N., MURAOKA K., TAKAHASHI H., MIURA M.: Two dimensional visual simulation of flames, smoke and the spread of fire. *Journal of Visualization and Computer Animation 5*, 1 (1994), 37–53.

[DEJ*99] DORSEY J., EDELMAN A., JENSEN H., LEGAKIS J., PEDERSEN H.: Modeling and rendering of weathered stone. In *Proc. of ACM SIGGRAPH 1999* (1999), pp. 225–234.

[DG96] DESBRUN M., GASCUEL M.-P.: Smoothed particles: a new paradigm for animating highly deformable bodies. In *Proc. of the Eurographics Workshop on Computer Animation and Simulation 1996* (1996), pp. 61–76.

[EMF02] ENRIGHT D., MARSCHNER S., FEDKIW R.: Animation and rendering of complex water surfaces. *ACM Transactions on Graphics (ACM SIGGRAPH 2002) 21*, 3 (2002), 736–744.

[Fed97] FEDKIW R.: *A Survey of Chemically Reacting, Compressible Flows*. PhD thesis, Dept. of Mathematics, Univ. of California, Los Angeles, 1997.

[FF01] FOSTER N., FEDKIW R.: Practical animation of liquids. In *Proc. of ACM SIGGRAPH 2001* (2001), pp. 23–30.

[FM97] FOSTER N., METAXAS D.: Modeling the motion of a hot, turbulent gas. In *Proc. of ACM SIGGRAPH 1997* (1997), pp. 181–188.

[FMP92] FOWLER D., MEINHARDT H., PRUSINKIEWICZ P.: Modeling seashells. In *Proc. of ACM SIGGRAPH 1992* (1992), pp. 379–387.

[FOA03] FELDMAN B., O'BRIEN J., ARIKAN O.: Animating suspended particle explosions. *ACM Transactions on Graphics (ACM SIGGRAPH 2003) 22*, 3 (2003), 708–715.

[FSJ01] FEDKIW R., STAM J., JENSEN H.: Visual simulation of smoke. In *Proc. of ACM SIGGRAPH 2001* (2001), pp. 23–30.

[Gat02] GATES W.: *Animation of Reactive Fluids*. PhD thesis, Dept. of Computer Science, The Univ. of British Columbia, 2002.

[GH04] GREENWOOD S. T., HOUSE D. H.: Better with bubbles: enhancing the visual realism of simulated fluid. In *Proc. of ACM SIGGRAPH/Eurographics on Computer Animation 2004* (2004), pp. 287–296.

[GSLF05] GUENDELMAN E., SELLE A., LOSASSO F., FEDKIW R.: Coupling water and smoke to thin deformable and rigid shells. *ACM Transactions on Graphics (ACM SIGGRAPH 2005) 24*, 3 (2005), 973–981.

[IKC04] IHM I., KANG B., CHA D.: Animation of reactive gaseous fluids through chemical kinetics. In *Proc. of ACM SIGGRAPH/Eurographics Symposium on Computer Animation 2004* (2004), pp. 203–212.

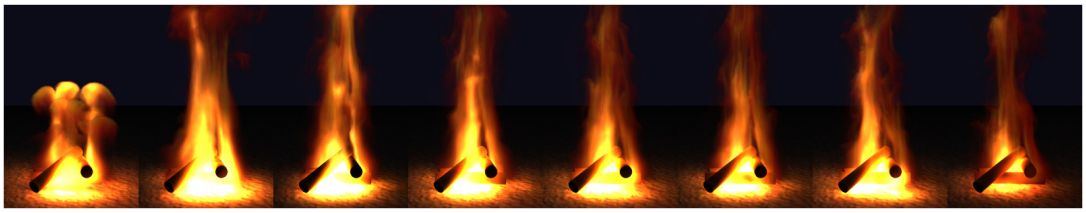
[KCC*06] KIM K., CHA D., CHANG B., KOO B., IHM I.: Practical animation of turbulent splashing water. In *Proc. of ACM SIGGRAPH/Eurographics Symposium on Computer Animation 2006* (2006), pp. 335–344.

[Lev02] LEVINE I.: *Physical Chemistry*, 5th ed. McGraw-Hill, ISBN 0-07-253495-8, 2002.

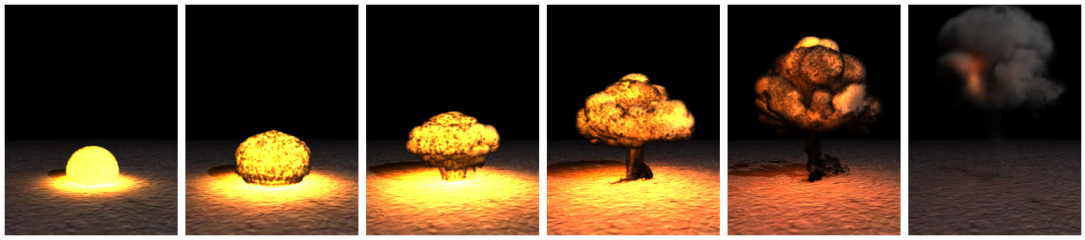
[LF02] LAMORLETTE A., FOSTER N.: Structural modeling of flames for a production environment. *ACM Transactions on Graphics (ACM SIGGRAPH 2002) 21*, 3 (2002), 729–735.

[LSF06] LOSASSO F., SELLE T., FEDKIW R.: Multiple interacting liquids. *ACM Transactions on Graphics (ACM SIGGRAPH 2006) 25*, 3 (2006), 812–819.

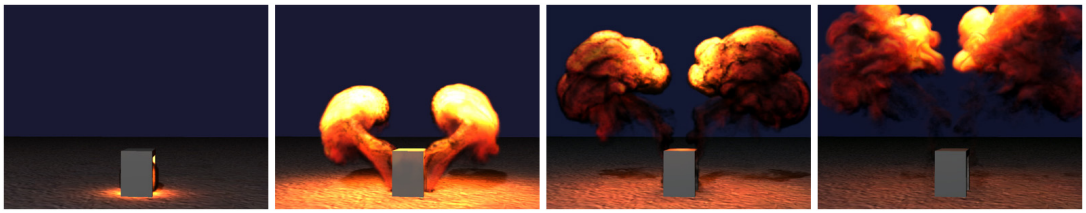
[LSSF06] LOSASSO F., SHINAR T., SELLE A., FEDKIW R.: Multiple interacting liquids. *ACM Transactions on Graphics (ACM SIGGRAPH 2006) 25*, 3 (2006), 812–819.



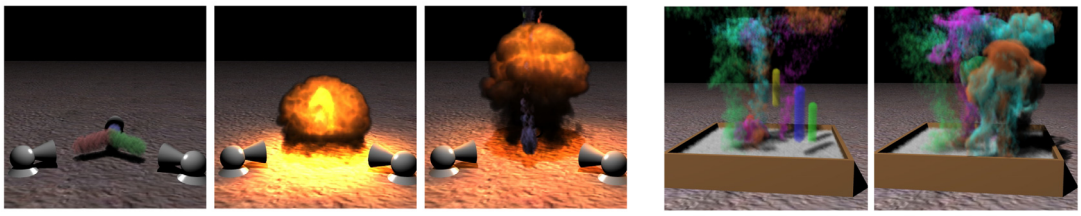
(a) Campfire scene



(b) Explosion scene 1

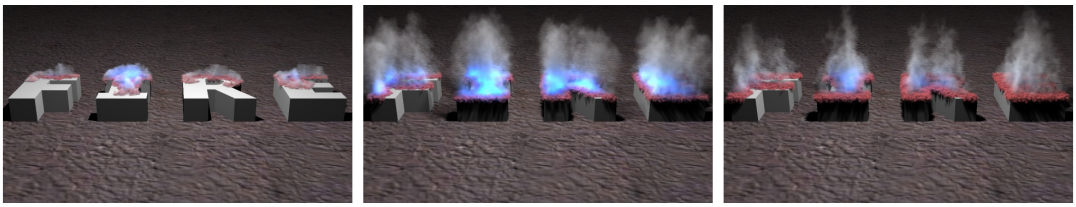


(c) Explosion scene 2



(d) Explosion scene 3

(e) Eleven species scene



(f) Catalysis scene 1

(g) Catalysis scene 2

Figure 5: *Example animation scenes.*

(a) Memory use per frame (MBytes)

	Grid resolution							
	$80 \times 80 \times 80$		$120 \times 120 \times 120$		$160 \times 160 \times 160$		$240 \times 240 \times 240$	
	Mean	Max	Mean	Max	Mean	Max	Mean	Max
Grid-only method	78.16	78.16	263.67	263.67	625.00	625.00	(2,109.37)	(2,109.37)
Hybrid method	39.14	40.25	131.89	135.02	312.57	318.94	1,054.83	1,056.08

(b) Number of used particles per frame

	Grid resolution							
	$80 \times 80 \times 80$		$120 \times 120 \times 120$		$160 \times 160 \times 160$		$240 \times 240 \times 240$	
	Mean	Max	Mean	Max	Mean	Max	Mean	Max
Material	415	3,999	413	3,997	412	3,998	418	3,998
Vortex	942	1,047	1,055	1,172	1,082	1,206	1,024	1,601
Soot	71,554	191,015	81,652	214,923	109,202	309,046	131,638	373,537

(c) Simulation time per frame (seconds)

		Grid-only method					Hybrid method					
		Step1	Step2	**Step3**	Step4	Total	Step1	Step2	**Step3**	**Step4**	Step5	Total
80^3	Mean	13.61	13.65	23.91	0.17	51.34	13.40	13.00	0.27	0.08	0.74	27.49
	Max	13.80	13.81	32.59	0.20	60.40	13.59	13.31	0.58	5.64	0.78	32.89
120^3	Mean	49.61	48.70	127.36	0.56	226.23	50.42	46.85	0.81	1.10	2.06	101.24
	Max	49.88	48.83	160.55	0.64	259.90	50.78	47.34	2.09	9.42	2.27	108.42
160^3	Mean	121.33	119.86	213.22	1.27	455.68	119.73	114.78	1.59	1.89	4.96	242.95
	Max	121.92	120.42	324.16	1.41	567.91	121.09	116.48	4.42	15.94	5.42	254.89
240^3	Mean	–	–	–	–	–	283.24	234.32	2.60	3.18	18.22	541.56
	Max	–	–	–	–	–	287.44	237.89	7.58	26.16	20.17	561.95

Table 2: *Performance comparison with a previous method. For the example scenario shown in Figure 5(b), our method ('Hybrid method') is compared to the previous method ('Grid-only method') [IKC04] with respect to various grid resolutions. In the grid-only method, the chemical reaction was applied in the grid-based third stage ('Step3'), while in the hybrid method, the corresponding task was carried out in the particle-based third and fourth stages ('Step3' and 'Step4'). In both spatial and temporal respects, our hybrid technique markedly outperformed the previous method that carried out all computations on the Eulerian grids. Omitted timings were unavailable because our PC with 2 GB of RAM was unable to allocate enough memory space (the estimated memory requirement is given in the first table).*

[MCG03] MÜLLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. In *Proc. of ACM SIGGRAPH/Eurographics Symposium on Computer Animation 2003* (2003), pp. 154–372.

[MMA99] MAZARAK O., MARTINS C., AMANATIDES J.: Animating exploding objects. In *Proc. of Graphics Interface 1999* (1999), pp. 211–218.

[MSKG05] MÜLLER M., SOLENTHALER B., KEISER R., GROSS M.: Particle-based fluid-fluid interaction. In *Proc. of ACM SIGGRAPH/Eurographics on Computer Animation 2005* (2005), pp. 237–244.

[NF99] NEFF M., FIUME E.: A visual model for blast waves and fracture. In *Proc. of Graphics Interface 1999* (1999), pp. 193–202.

[NFJ02] NGUYEN D., FEDKIW R., JENSEN H.: Physically based modeling and animation of fire. *ACM Transactions on Graphics (ACM SIGGRAPH 2002) 21*, 3 (2002), 721–728.

[OH95] O'BRIEN J., HODGINS J.: Dynamic simulation of splashing fluids. In *Proc. of Computer Animation 1995* (1995), pp. 198–206.

[PK05] PARK S., KIM M.: Vortex fluid for gaseous phenomena. In *Proc. of ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2005), pp. 261–270.

[PTB*03] PREMOŽE S., TASDIZEN T., BIGLER J., LEFOHN A., WHITAKER R. T.: Particle-based simulation of fluids. *Computer Graphics Forum (Eurographics 2003) 22*, 3 (2003), 401–410.

[RNGF03] RASMUSSEN N., NGUYEN D., GEIGER W., FEDKIW R.: Smoke simulation for large scale phenomena. *ACM Transactions on Graphics (ACM SIGGRAPH 2003) 22*, 3 (2003), 703–707.

[SF95] STAM J., FIUME E.: Depicting fire and other gaseous phenomena using diffusion processes. In *Proc. of ACM SIGGRAPH 1995* (1995), pp. 129–136.

[Sim90] SIMS K.: Particle animation and rendering using data parallel computation. In *Proc. of ACM SIGGRAPH 1990* (1990), pp. 405–413.

[SRF05] SELLE A., RASMUSSEN N., FEDKIW R.: A vortex particle method for smoke, water and explosions. *ACM Transactions on Graphics (ACM SIGGRAPH 2005) 24*, 3 (2005), 910–914.

[TFK*03] TAKAHASHI T., FUJII H., KUNIMATSU A., HIWADA K., SAITO T., TANAKA K., UEKI H.: Realistic animation of fluid with splash and foam. *Computer Graphics Forum (Eurographics 2003) 22*, 3 (2003), 391–400.

[TOT*03] TAKESHITA D., OTA S., TAMURA M., FUJIMOTO T., MURAOKA K., CHIBA N.: Particle-based visual simulation of explosive flames. In *Proc. of the 11th Pacific Conference on Computer Graphics and Applications* (2003), pp. 482–486.

[Tur91] TURK G.: Generating textures on arbitrary surfaces using reaction-diffusion. In *Proc. of ACM SIGGRAPH 1991* (1991), pp. 289–298.

[WK91] WITKIN A., KASS M.: Reaction-diffusion textures. In *Proc. of ACM SIGGRAPH 1991* (1991), pp. 299–308.

[WLMK02] WEI X., LI W., MUELLER K., KAUFMAN A.: Simulating fire with texture splats. In *Proc. of IEEE Visualization 2002* (2002), pp. 227–234.

[YOH00] YNGVE G., O'BRIEN J., HODGINS J.: Animating explosions. In *Proc. of ACM SIGGRAPH 2000* (2000), pp. 29–36.

[ZB05] ZHU Y., BRIDSON R.: Animating sand as a fluid. *ACM Transactions on Graphics (ACM SIGGRAPH 2005) 24*, 3 (2005), 965–972.