Computers & Graphics 76 (2018) 107–116

Contents lists available at ScienceDirect

Computers & Graphics

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

Special Section on SIBGRAPI 2018

Heat-based bidirectional phase shifting simulation using position-based dynamics

Steeven Villa Salazar[a,1,*], Jose Abel Ticona[a,1], Rafael Torchelsen[b], Luciana Nedel[a], Anderson Maciel[a]

[a] Universidade Federal do Rio Grande do Sul (UFRGS) Av. Bento Gonçalves, Porto Alegre 9500, Brasil
[b] Universidade Federal de Pelotas (UFPel) R. Gomes Carneiro, 1, Pelotas, Brasil

ARTICLE INFO

Article history:
Received 1 May 2018
Revised 4 September 2018
Accepted 8 September 2018
Available online 12 September 2018

Keywords:
Position-based dynamics
Smoothed particle hydrodynamics
Phase transition
Thermal phenomena

ABSTRACT

Phase-change phenomena are present in our daily life. Examples are the evaporation of a fluid when it reaches its boiling temperature, the condensation of water vapor in air due to the pressure changes or due to the difference of temperature in boundaries, and the melting of snow when winter is ending. Current development in physics-based animation allows the simulation of these phenomena, but an integrated solution for modeling bidirectional phase-shifting objects is not available for games and other virtual environments. In this work we present a temperature-based method that drives phase transition phenomena based on latent heat of materials using position-based dynamics (PBD). Modifications to density, viscosity and distance PBD constraints are proposed to simulate the necessary thermal phenomena. Results show that melting, fusion, evaporation, condensation, dilation and even convection effects can be obtained by modifying the original PBD constraints in function of latent heat.

1. Introduction

Physically based animation has increased its importance in animation movies, the new generation of games and immersive virtual environments in the later years. Capturing real-world physical behaviors is a continuous objective widely pursued by the computer graphics community. Phase transition phenomena have long represented an exciting challenge to study in the context of animation. The cost to simulate the complex non-linear physical laws involved is prohibitive for interactive animation, in such a way that previous methods focused on off-line simulation [1]. It is necessary to build a heat-based model to govern the phase change phenomena and, furthermore, simplify the thermodynamic model to a temperature-based model to efficiently control the different phase transitions [2]. Moreover, to correctly simulate real-world situations, an object should account for the thermodynamic properties of its composing material when transitioning between solid, liquid and gas phases. Classic methods such as Finite Elements were used to computationally model these natural phenomena. Unfortunately, those models entail high computational cost due to the complex calculations involved.

Several models have been developed to better model the energy transport within this thermal problem (melting, fusion, evaporation, condensation) [2–5]. Nonetheless, most of those works do not provide a general solution and focus on resolving specific parts instead of modeling the whole problem, usually covering one or two of the transitions. A common approach to achieve thermal and phase change simulations is to combine some of the existing techniques used to represent fluids (Smoothed Particles Hydrodynamics (SPH), Lattice Boltzmann Method (LBM), Fluid Implicit Particles (FLIP), Particle In Cell (PIC)) with techniques used to simulate solid bodies (Mass Springs systems (MSD), Finite Elements (FEM), Position-based Dynamics (PBD)). Those methods are not fully compatible and need special procedures to be coupled properly.

In this paper, we propose a full Lagrangian model to simulate phase transitions between solid, liquid and gas states. Our model is sufficiently general to allow a continuous representation of all the basic first-order phase transitions. We based our model on the Position-based dynamics (PBD) framework [6] due to its proven stability and its capacity to simulate physical phenomena with significantly lower computational cost than methods based on differential equations. Using this framework, and inheriting its functionality and robustness, we can handily couple all the states modifying the existing constraints that model solids and fluids in order to obtain phase-shifting effects. We demonstrate that just by using PBD and the methods previously demonstrated within that framework (for instance, SPH to calculate density constraint), it is

* Corresponding author.
E-mail address: dsvsalazar@inf.ufrgs.br (S. Villa Salazar).
[1] Both authors contributed equally to this work.

https://doi.org/10.1016/j.cag.2018.09.004
0097-8493/

possible to achieve plausible results when replicating thermodynamic phenomena.

The remainder of this paper is structured as follows: first, we review previous works on simulation of phase shifting and position-based simulations (Section 2). Next, in Section 3, we explain the techniques used to obtain our simulation. We detail the modifications proposed to existing methods and how we handle solids, liquids and gases and the heat transfer model used in our model. Section 4 describes our algorithms to couple different phases and transitions. We introduce how we manage the PBD constraints and thermal properties of the particles. Finally, we present the results achieved and discuss our main findings.

2. Related works

Melting and fusion. In 2014, Stomakhin et al. [3] published a work addressing melting, solidifying, and heat transfer using fluid implicit particles (FLIP) and Material Point Method (MPM) to simulate fluid particles. They obtain those effects varying material properties in function of temperature and phase, but their work does not address evaporation or condensation. Similarly, Gao et al. [7] reported a work coupling fluid–solid phases using PBD and FLIP. However, they covered only melting simulation. In another work, Gao et al. [2] extended their method to include gas, but they do not take into account condensation. Gao and colleagues were looking for a general solution to integrate the main phase-transition and states of the matter. We pursue very similar goals. However, we follow a different approach. The main difference is our fluid model. While they used a combination of Lagrangian and Eulerian approaches, we propose a unified fluid model to all states of matter. Very recently, Weiler et al. [8] developed an implicit viscosity solver for Smoothed Particle Hydrodynamics (SPH) fluids capable to represent high viscous fluids consistently. Despite their work is mainly focused on viscosity handling, they extended the approach to simulate melting.

Evaporation and condensation. Fluid-fluid interaction was modeled by Müller et al. [9]. They reported a technique to simulate vaporization effects depending on the temperature of particles. Their work does not take into account other phase transitions and does not use latent heat as a criterion to induce the phase change. More recently, Hochstetter and Kolb [4] published a paper simulating evaporation and condensation transitions. Their work is mostly focused on the visual effects. They used SPH as well as the Cleary and Monaghan Heat model [10] to manage particle temperatures. This is one of the models that we implemented in this work (see Section 3.5). Ren et al. [11] worked in a method to simulate gas using SPH and performing calculation just in the visible particles, unlike the traditional technique that consists on filling the domain with transparent air particles to generate convective forces. With the exception of the aforementioned papers, few works address both condensation and evaporation transitions using Lagrangian models.

Fluid–solid modeling. SPH is one of the most widespread frameworks to simulate fluids [12]. To enforce incompressibility in SPH, Solentanthaler and Pajarola [13] used a predictive-corrective approach, while Ihmsen et al. [14] presented an implicit incompressible SPH. Another relevant work addressing this topic was done by Becker and Teschner [15], where they introduced the widely used Weakly Compressible SPH. In 2013, Macklin and Müller [16] introduced the Position-based fluids, which is an implementation of SPH into the PBD Framework. Their approach helps to obtain an incompressible fluid at relatively low computational cost, consequently allowing large timesteps. Keiser et al. 2005 [17] modeled solid and fluids merging Navier Stokes equations of movement into a Lagrangian framework. Despite SPH being mainly used to represent fluid models, the techniques have also been applied to model soft bodies [18,19].

In recent years, Position-based dynamics has become more and more popular. In 2014, Macklin et al. [20] reported a "Summary" work demonstrating the versatility of PBD to simulate solids, fluids, plastic deformations, soft solids, and cloth. There is, nevertheless, a gap in the literature that our methods here presented aim to fill: providing the means to simulate plausible phase-shifting materials into a full Lagrangian framework.

3. Methods

In this section, we introduce a novel combination of methods and technologies into an integrated solution to cover all state changes in the same framework. In this sense, we chose and take advantage of the Position-based dynamics framework and its inherent benefits to model solid bodies together with the Position-based fluid approach to model liquid and gas. We then combine PBD and PBF particles with an SPH-based temperature model to track heat flow among them. For the sake of clarity, we marked our equations with a star ($\star$) beside the equation number to differentiate them from the other equations previously available in the literature.

3.1. Extended position based dynamics

Originally developed by Müller and Heidelberg in 2007 [6], PBD is governed by a set of constraints C that must satisfy an equality (or inequality), i.e. $C_i(x+\Delta x)=0$, or its linearized representation by a series of Newton steps:

$$C_i(x+\Delta x) \approx C_i(x) + \nabla C_i(x)\Delta x. \tag{1}$$

where $x=[x_1, x_2, \ldots, x_1]^T$ is the vector of positions. Eq. (1) is centered around x and Δx is restricted to take values along the constraint gradient $\nabla C_i(x)$ [20]. In contrast with the regular PBD, in extended PBD (XPBD) [21], the scaling factor λ used to conserve the linear and angular momenta is a delta. In that way, the position correction is given by Eq. (2):

$$\Delta x = \nabla C_i(x) w_i \Delta\lambda. \tag{2}$$

Where w_i represents the inverse masses and $\Delta\lambda$ is given by:

$$\Delta\lambda = -\frac{C_i(x) - \lambda\tilde{\alpha}}{\sum_j w_j \left|\nabla C_j(x)\right|^2 + \tilde{\alpha}}. \tag{3}$$

With $\tilde{\alpha}=\alpha/\Delta t^2$ and α the compliance of the material. λ needs to be stored in each iteration:

$$\lambda_{i+1} = \lambda_i + \Delta\lambda. \tag{4}$$

Solid body dilation. Young's modulus ($E=\sigma/\epsilon$) defines the linear relationship between applied stress (σ) and associated deformation (ϵ) [22].

Elastic properties of solids can be introduced easily by extending the standard distance constraint of PBD [23]. We adapt this parameter as follows.

The more the temperature increases, the more the solid body dilates. Consequently, E must decrease as it represents the solid's rigidity:

$$E = \frac{2E_0}{1+e^{\Delta T}}. \tag{5}$$

where E_0 is the Young's modulus of the material, a given property mostly obtained empirically. In addition, we establish the initial distance between particles to depend on the linear coefficient

of expansion (e) and ΔT. This set up admits both dilation and contraction, depending on the direction of the heat change:

$$\Delta d_0 = e\Delta T d_0. \tag{6}$$

Hence, we obtain the final position correction for each couple of particles by adding (5) and (6) to the original corrections $C(x_i, x_j) = | x_i - x_j | - d_0$:

$$\begin{aligned} \Delta x_1 &= -E\frac{w_1}{w_1 + w_2 + \tilde{\alpha}}[| x_{1,2} | - d_0(1 + e\Delta T) - \lambda\tilde{\alpha}] \\ \Delta x_2 &= +E\frac{w_2}{w_1 + w_2 + \tilde{\alpha}}[| x_{1,2} | - d_0(1 + e\Delta T) - \lambda\tilde{\alpha}] \end{aligned} \quad \star. \tag{7}$$

Where d_0 is the initial distance.

3.2. Smoothed particles hydrodynamics

SPH can be seen as an interpolation scheme between scattered particles that acts in a given radius around a point centered in $\mathbf{r}$:

$$A(\mathbf{r}) = \sum_j m_j \frac{A_j}{\rho_j} W\big(\mathbf{r} - \mathbf{r}_j, h\big). \tag{8}$$

Here m_j is the mass of the neighbor particle j, ρ_j represents its density, and W is a smoothing kernel with support radius $2h$. So, all the summations are performed over all particles j within the radius of the kernel. With exception of the kernel, all parameters are scalars when evaluated in each particle.

Notice that, for the 3D case, while $W(r - r_j, h)$ is a scalar field $(\mathbb{R}^3 \to \mathbb{R})$, its gradient $\nabla W(r - r_j, h)$ is a vector field $(\mathbb{R}^3 \to \mathbb{R}^3)$. So, the gradient of the function A is obtained just by differentiating the interpolation kernel [10]. In that way, we could use Eq. (8) to evaluate density. The following result would be obtained:

$$\rho_i = \sum_j m_j W(\mathbf{r}_i - \mathbf{r}_j, h). \tag{9}$$

This means that particle density relies just on its mass and the influence of its neighborhood. In this paper, we work with two kernels, in the same way as Muller et al. [24]. Poly6 Kernel is used to compute densities and spiky Kernel to calculate gradients.

Viscosity modeling. To damp out non-physical oscillations, we use the XSPH viscosity approach of Schechter and Bridson [25] due to its low cost and easiness to manipulate in contrast with the classic XSPH. In this manner, we damp out the noise when updating velocities:

$$\mathbf{v}_i^{next} = \mathbf{v}_i + \mu \sum_j (\mathbf{v}_i - \mathbf{v}_j) W\big(\mathbf{r} - \mathbf{r}_j, h\big). \tag{10}$$

μ is tunable. We use this parameter in further sections to generate convection effects.

3.3. Position based fluids

One of the main issues when working with SPH is the incompressibility, and several works have addressed this problem in the past. Works such as Shao and Edmond [26], Hu and Adams [27] and Solenthaler and Pajarola [13] notably improved this SPH weakness, but the requirement to have a small time-step to guarantee stability remains. A feasible solution to allow larger time-steps was presented by Muller and Macklin in 2013 [16]. They calculate the fluid particle density as another constraint in the PBD framework:

$$C(\mathbf{x}) = \frac{\rho_i}{\rho_0} - 1. \tag{11}$$

Note that ρ_i can be computed from the SPH density equation (Eq. (9)) and ρ_0 is the rest density of the fluid. This constraint could be projected in the same fashion as shown by the distance constraint. It is recommended to use constraint force mixing [28] to regularize the constraint. This leads λ to be:

$$\lambda = -\frac{C_i(x)}{\sum_j \left|\nabla C_j(x)\right|^2 + \varepsilon}. \tag{12}$$

being ε a relaxation constant [16] specified by the user at the beginning of the simulation.

3.4. Solid–fluid–gas coupling

To handle boundary interactions, we calculate the density number $\delta_i = \sum_k W_{ik}$ according to the approach of Akinci et al. [29]. Solid influence is taken into account while calculating the fluid densities. It is important to highlight that we correct the solid particle position by applying the density constraint to them. In such way, all phases interact in the same framework. Moreover, gas is treated as liquid-fluid particles but changing its internal properties. This will be discussed further in the section about transition coupling. The equation below shows how density calculation is performed:

$$\rho_i = \sum_j m_j W_{ij} + \sum_k m_k W_{ik} + \sum_b \frac{\rho_0}{\delta_b} W_{ib}. \tag{13}$$

where j particles are fluid (liquid and gas phase), k represents solid particles and b are boundary particles. So, i could be either a solid or a fluid particle. Using this approach, we can obtain solid–fluid–gas interaction in a straightforward fashion.

3.5. Heat transfer

Temperature is the core variable in this work since phase change phenomena occur when a specific material reaches its temperature threshold (we will extend this assertion later in this section). In a homogeneous medium, heat transfer is given by [4]:

$$\frac{dT}{dt} = \frac{\nabla(k\nabla T)}{\rho c_p}. \tag{14}$$

The literature offers several SPH-oriented models using **heat** transfer. One of them is an explicit scheme to represent the time discretization of the heat equation (Eq. (14)). Besides, a 1999 work by Cleary and Monaghan [10] is still today one of the most popular models used to simulate **temperature**-based phenomena in SPH. So, we implemented both models to provide and track temperature distributions among our position-based particles:

Explicit model. This is, perhaps, the simplest way to discretize Eq. (14) to SPH:

$$\left(\frac{dT}{dt}\right)_{i,j} = 2\phi \sum_j \frac{m_j}{\rho_j}\big(T_j - T_j\big)\nabla W_{ij}. \tag{15}$$

Note that, in this equation, j is a neighbor particle independently of its current phase (excluding boundaries that have no phase information). The parameter ϕ is a scale factor to handle the amount of temperature transported among particles.

As boundary particles have no temperature properties, we handle this region in the following way:

$$\left(\frac{dT}{dt}\right)_{i,b} = 2\phi \sum_b \frac{\rho_0}{\delta_b}\big(T_j - T_b\big)\nabla W_{ib}. \tag{16}$$

This expression controls the influence of all boundary b particles around i. The total temperature in a single particle is given by the sum of the influences of the neighboring particles and the boundary heat, induced using Eq. (16). This can be expressed as follows:

$$\left(\frac{dT}{dt}\right)_i = \left(\frac{dT}{dt}\right)_{i,j} + \left(\frac{dT}{dt}\right)_{i,b}. \tag{17}$$

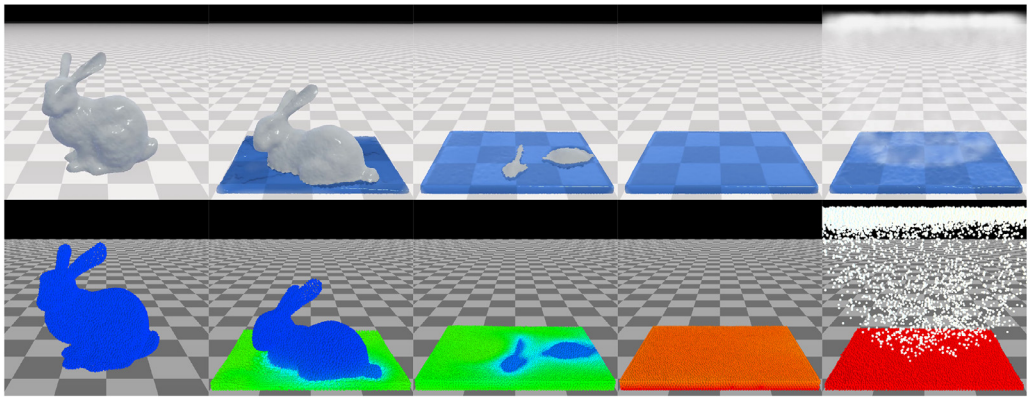

Fig. 1. Melting and evaporation process with 27 K particles involving three states and two phase changes: In the left image, a solid bunny is shown. When temperature increases at the bottom, the bunny starts to melt. After that, all particles are liquid and continue to heat. Finally, liquid particles switch to gas when the latent heat of evaporation is reached.

Cleary and Monagham Model. C&M approach is an important reference on heat transfer related to SPH. They modeled heat transfer taking into account the specific heat capacity (c) and thermal conductivity (k), allowing for different conduction behaviors [10]. They arrived at the following equation:

$$\left(\frac{dT}{dt}\right)_{i,j} = \frac{V_i}{m_i c_i} \sum_j \frac{4k_i k_j}{k_i + k_j} V_j (T_i - T_j) \nabla W_{ij}. \tag{18}$$

In our modified model, we handle the boundaries by introducing fixed temperatures when evaluating boundary particles:

$$\left(\frac{dT}{dt}\right)_{i,b} = \frac{V_i}{m_i c_i} 2k_i \sum_b \frac{\rho_0}{\delta_b} (T_i - T_b) \nabla W_{ib}. \tag{19}$$

This allows us to induce temperatures coming from the boundaries.

3.6. Latent heat

At the beginning of the previous section, we stated that phase change occurs when a specific material reaches its critical temperature. It is natural to think that a phase change depends on the temperature value and this is not completely wrong. The point is that, when the phase change is happening, the temperature remains constant. So, it is not the variation in temperature but the variation in energy that makes phase-change to occur.

Let us define temperature as sensible heat, the heat we can measure with a thermometer. Latent heat, in turn, is the energy used to perform the phase change. We illustrate this in Fig. 2. In the initial (low temperature) state, a small amount of energy is stored. As the temperature increases (sensible heat), accumulated heat also increases. When the body meets a critical temperature (melting and boiling points), accumulated heat (latent) still increases but the temperature remains constant.

This means that the sensible heat stopped increasing. After gaining enough energy, i.e. after accumulating enough latent heat, the material reaches its latent heat of fusion/vaporization threshold and transitions to the next state of matter. Then, the sensible heat starts increasing again. This cycle is repeated in any of the phase changes and works in both ways. Notice that the latent heat quantity required to melt some material is rarely the same that is required to boil.

Computational implementation. To handle states and transitions, we store current state and last state depending on the particle

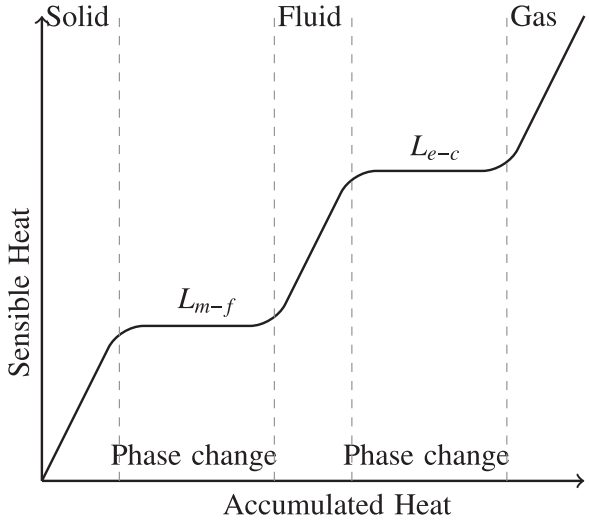

Fig. 2. Relation between heat and phase change.

temperature and the critical temperature values:

$$State = \begin{cases} Solid & T_i < T_m, \\ Phase\ change_{m-f} & T_i = T_m \\ Fluid & T_m < T_i < T_e. \\ Phase\ change_{e-c} & T_i = T_e \\ Gas & T_i > T_e \end{cases} \tag{20}$$

Sub-index $m-f$ and $e-c$ means $melting-fusion$ and $evaporation-condensation$ respectively. Here we cover all the possible states of our particles and, if a given particle is shifting its state, the temperature must be constant. The model saves the latent heat of each phase change independently: L_{m-f} for latent heat of melting and L_{e-c} for latent heat of evaporation. The rate of change in latent heat is calculated from the change in temperature and specific heat as:

$$\frac{dL}{dt} = c_i \Delta T. \tag{21}$$

with ΔT obtained from the heat model. Next, we increment latent heat applying Algorithm 1.

Our heat scheme works in the following way: when the phase change is occurring, which is to say $State = Phase\ change_{m-f}$ or $State = Phase\ change_{e-c}$, latent heat starts increasing or decreasing based on Eq. (21), and then we enforce latent heat values to be within boundaries $[0, L_{threshold}]$. Other cases check if future tem-

Algorithm 1 Latent heat update.

1: **switch** *State Particle i* **do**
2: **case** *Phase change*$_{m-f}$
3: $L_{m-f} \leftarrow L_{m-f} + \Delta L$(Using Eq. (21))
4: **if** $L_{m-f} < 0$ **then**
5: $L_{m-f} \leftarrow 0$
6: **else if** $L_{m-f} >= L_{m-f,threshold}$ **then**
7: $L_{m-f} \leftarrow L_{m-f,threshold}$
8: **case** *Phase change*$_{e-c}$
9: $L_{e-c} \leftarrow L_{m-f} + \Delta L$(Using Eq. (21))
10: **if** $L_{e-c} < 0$ **then**
11: $L_{e-c} \leftarrow 0$
12: **else if** $L_{e-c} >= L_{e-c,threshold}$ **then**
13: $L_{e-c} \leftarrow L_{e-c,threshold}$
14: **case** *Solid*
15: **if** $T_i^{n+1} \geq T_{m-f}$ **then**
16: $State_i \leftarrow$ *Phase change*$_{m-f}$
17: **case** *Fluid*
18: **if** $T_i^{n+1} \leq T_{m-f}$ **then**
19: $State_i \leftarrow$ *Phase change*$_{m-f}$
20: **else if** $T_i^{n+1} \geq T_{e-c}$ **then**
21: $State_i \leftarrow$ *Phase change*$_{e-c}$
22: **case** *Gas*
23: **if** $T_i^{n+1} \leq T_{e-c}$ **then**
24: $State_i \leftarrow$ *Phase change*$_{e-c}$

peratures are out of the current state and assign a phase change identifier to the particle.

4. Transition coupling

Latent heat is the driver of all phase changes. Transition coupling is performed as shown in Algorithm 2. Further details will be given afterwards. Let us now overview the main loop. We start initializing variables in lines 2–4 for each particle. The values of these variables depend on material properties, and interactions between different materials are allowed. In line 6, positions are calculated according to the position based dynamics loop. Next (line 8), gravity values are assigned depending on the latent heat of vaporization. From lines 11–15 all constraints are solved, then in line 17, we apply an algorithm to manage distance constraints on solid particles. Finally, we perform the heat transfer among particles. The last three lines are the updates on changed values.

Algorithm 2 Main loop.

1: **for all** Particles **do**
2: $x_i \leftarrow 0, v_i \leftarrow 0, w_i \leftarrow \frac{1}{m_i}$
3: $T_i \leftarrow T_0, L_m \leftarrow 0, L_e \leftarrow 0$
4: Set $T_m, T_e, L_{m,threshold}, L_{e,threshold}, c_i, k_i, \phi_i$
5: **loop**
6: **for all** Particles **do** Calculate Next Positions
7: **for all** Fluid Particles **do**
8: Set Gravity Acceleration(L_e)
(Using Eq. (22))
9: **for all** Particles **do** Neighborhood Search
10: **loop**Iterations:
11: **for all** Particles **do**
12: Density Constraint(L_e)
(Using Eqs. (11) and (23))
13: **for all** Fluid Particles **do** Viscosity Constraint
(Using Eq. (10))
14: **for all** Solid Particles **do**
15: Distance Constraint(T_i)
(Using Eq. (10))
16: **for all** Particles **do**
17: Manage Constraints(L_m)
(Using Algorithm 3)
18: **for all** Particles **do** Heat Transfer
(Using Eq. (17) and algorithm 1)
19: **for all** Particles **do** Update Temperatures
20: **for all** Particles **do** Update Velocities
21: **for all** Particles **do** Update Neighborhood

4.1. Solid–liquid phase

When in the solid state, material properties are governed by the temperature value. Young's modulus E and α values are modified according to the current temperature (see Eqs. (6) and (5)) until it meets the melting point. There, while the latent heat increases, both values remain constant. After that, the material attains the latent heat of fusion and its maximal thermal expansion, and consequently, the phase change occurs:

Solid–liquid (melting). Solid particles have distance constraints to conserve the solid's shape. Each particle represents a node in the solid and each constraint involves two particles. As a particle can admit more than one constraint, if its latent heat reaches the critic heat, all constraints containing this particle are deleted.

Liquid–solid (freezing). In contrast with melting, freezing (or solidification) does the opposite process. Here, we create new constraints based on a support radius rs_{max} (not necessarily the same used with the SPH kernels). When two neighboring fluid particles are decreasing their latent heat, they are candidates for solidification. If both meet the solidification critical value, a new distance constraint is generated. This process stops as soon as the particle is out of the *Phase change*$_{m-f}$ status. Notice that, for a particle to solidify, its last state must be *fluid*. Additionally, if a particle reaches its maximum number of constraints nC_{max}, no new constraint are created for this particle. Algorithm 3 is introduced to manage the life cycle of distance constraints.

Algorithm 3 Manage constraints.

1: **for all** Distance Constraint Particles i,j **do**
2: **if** $State_i =$ *Phase change*$_{m-f}$ & $State_j =$ *Phase change*$_{m-f}$ **then**
3: Delete Distance Constraint i,j
4: **for all** Particles i **do**
5: **if** $State_i =$ *Phase change*$_{m-f}$ **then**
6: **for all** Neighboring Particle j **do**
7: *Cond* 1 $\leftarrow State_j <$ *Phase change*$_{m-f}$
8: *Cond* 2 $\leftarrow | x_i - x_j | < r_{max}$
9: *Cond* 3 $\leftarrow nC_i < nC_{max}$
10: **if** *Cond* 1 & *Cond* 2 & *Cond* 3 **then**
11: new Distance Constraint i,j

4.2. Liquid–gas phase

To obtain vaporization, we modify the initial rest density ρ_0 in the density constraint (Eq. (11)) multiplying by a scaling function β. Doing this, convection behavior similar to water boiling arises when latent heat is increasing. Eq. (22) represents an inverse sigmoid function:

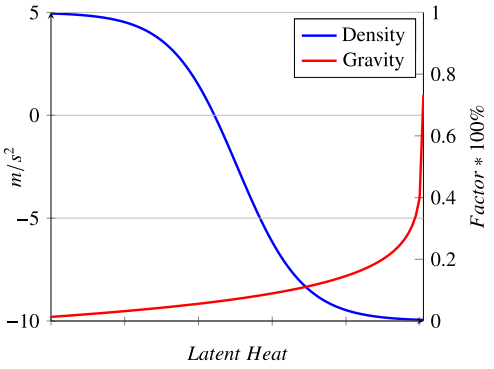

Fig. 3. Phase functions: Buoyancy of a particle is given by this function. This allows us to simulate ebullition and evaporation effects. Blue: (Right scale) The density factor decreases as latent heat increases, providing a natural convection alike effect. Red: (Left scale) Gravity remains low in almost the 90 percent of the phase change, letting the convection effect occur. Gravity arises when the density is low generating the evaporation effect. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

$$\beta(\gamma) = 1 - \frac{1}{1 + \left(1 + \frac{12}{L_{e,threshold}}\right)^{\gamma}} \quad \star. \tag{22}$$

with $\gamma = \frac{L_{e,threshold}}{2} - L_e$. The behavior of Eq. (22) is shown in Fig. 3. Moreover, vaporization simulation is performed setting the particle gravity as a function of the latent heat. The function g (shown in Eq. (23)) keeps the particle gravity unchanged in smaller values of latent heat and, as it increases, gravity turns to positive, producing the vaporization effect (see Fig. 3):

$$g(\omega) = \frac{10.8}{\ln\left(\frac{L_{e,threshold}+0.02}{0.02}\right)} \ln(\omega) - 9.8 \quad \star. \tag{23}$$

with $\omega = \frac{L_{e,threshold}+0.02}{L_{e,threshold}+0.02-L_e}$.

5. Results

We implemented our model in C++ with the Eigen library. We used a 3.40 GHz i5-4670 CPU with 16 GB RAM to generate all results here presented. The whole implementation is sequential on the CPU for the sake of simplicity and reproducibility. Besides, we used post-processed rendering to better isolate the software elements for analysis. The Nvidia Flex environment was used for graphics rendering.

To assess the capabilities of our method, we developed scenarios that cover the effects of all phase transitions.

5.1. Scenarios

Melting and vaporization. Fig. 1 illustrates the melting and vaporization processes with 27k particles. We set a fixed high temperature on the floor where the solid-cold bunny is placed sitting.

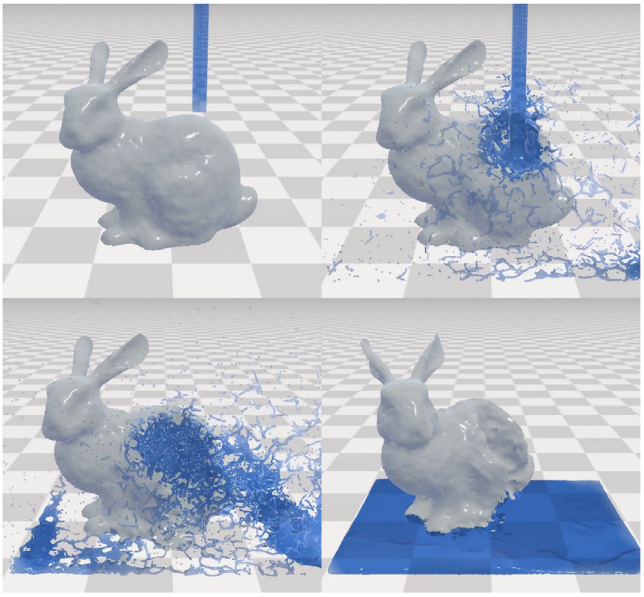

Fig. 5. Melting simulation: Hot water drops over an ice-bunny that melts when hot particles contact with cold particles. Interestingly, some liquid particles cool enough to freeze when in contact with the very cold ice particles.

When a particle meets the latent heat threshold, our constraint manager breaks all the connections among this particle and its neighbors. When the particles change their states to liquid, the viscosity constraint is activated. Later, the transition to gas completes the process. This scenario shows one way of the whole process, involving three states: solid, liquid, and gas, as well as the two phase changes.

Solid and fluid interactions. When a cold solid is thrown into a warm liquid, or vice-versa, the heat-transfer phenomena proceeds to eventually modify the phase of some of the particles in both the liquid and the solid, e.g. the interaction between ice and warm water. A higher amount of energy is present in the hot liquid particles. Then, energy is transported from the liquid, let us say, water, to the cold solid, melting it and decreasing the sensible heat present in the remaining water. Fig. 5 depicts a scenario where some amount of hot water falls over an icy bunny, melting the ice partially in the regions affected with enough energy to break the local solid constraints. When the falling particles do not have enough energy to break the constraints, however, a different phenomenon takes place. In the scenario of Fig. 7, instead of melting the bunny, some liquid particles eventually solidify when they lose enough heat.

Natural convection. The use of sigma and logarithmic functions allows us to simulate convective effects on fluids. In a boiling water scenario, the change in density due to the increase of latent heat causes particles with higher energy to move up and particles with

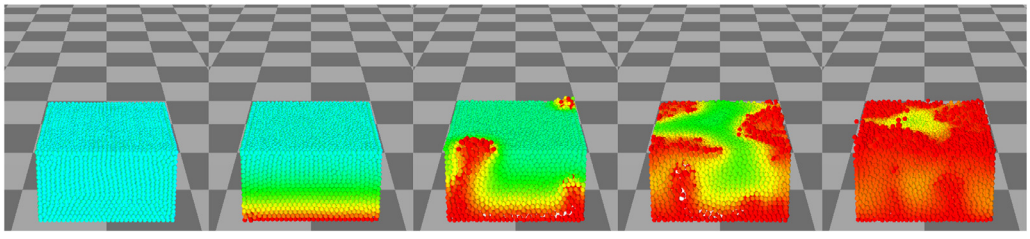

Fig. 4. Natural convection effect.

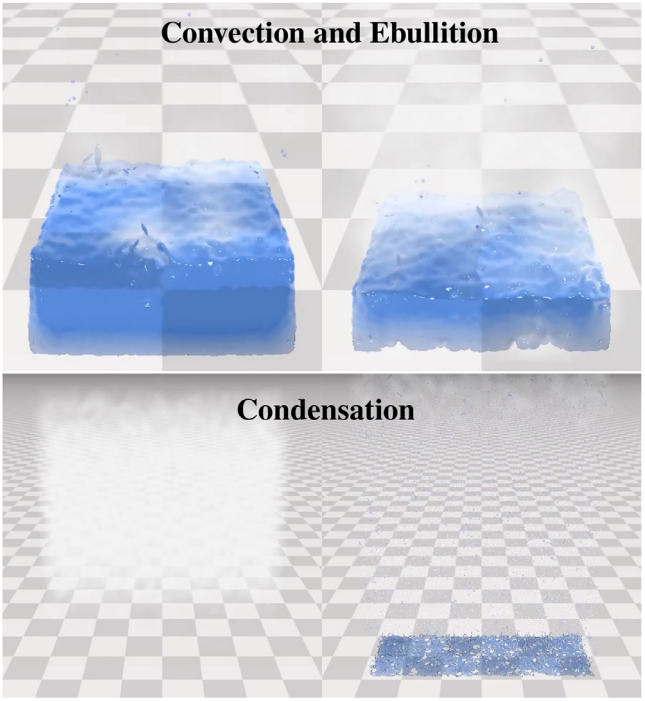

Fig. 6. Ebullition (top) and Condensation (bottom) effects. Water in ebullition, with 50k particles, demonstrates the effect of vaporization combined with convection that is typical of boiling water. With 26k particles, condensation illustrates the cloud formation and precipitation (rain).

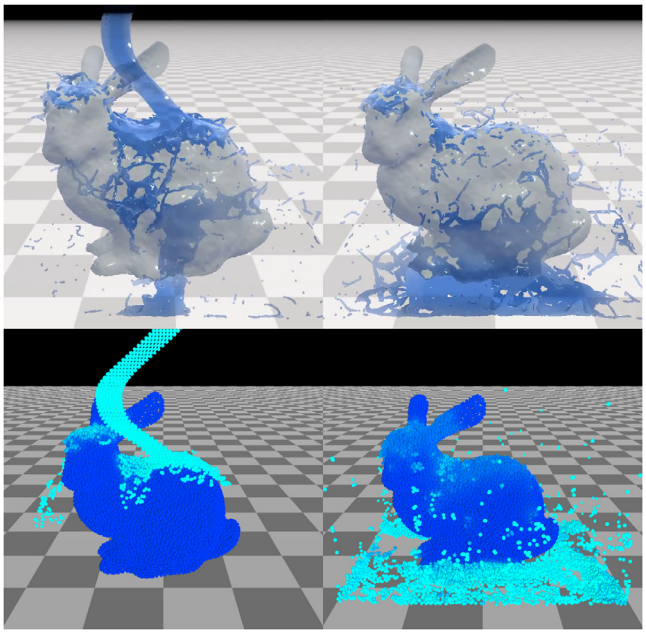

Fig. 7. Solidification simulation: Warm water falls over the icy bunny but the fluid particles have not enough heat to melt it. Then some fluid particles lose enough of their heat to freeze.

less energy to move down. In Fig. 6 a boiling effect is shown in the top frame. When lower particles absorb enough energy, they rise to the surface and eventually vaporize. Fig. 4 shows explicitly how particles with lower density and higher temperature arises from the bottom to the surface of the container.

Vaporization & condensation. In a similar way to the convection scenario, we use the σ function to simulate vaporization. The buoyancy of gas particles depends on their latent heat of evaporation. The more latent heat of evaporation a particle has, the more buoyancy it experiences. Other works such as the ones proposed by Muller et al. and Macklin et al. [9,20] suggest the introduction of air particles in the free domain to generate buoyancy and let the gas particles raise. Such approach could help to better model the fluid-air interaction in terms of heat transfer. However, its computational efficiency is questionable under some circumstances. Fig. 1, in its last two frames, shows our evaporation process in action: fluid particles start a convective-like effect and when enough energy is available, buoyancy effects begin raising these particles based on Eq. (23), as Fig. 3 illustrates. Then, in Fig. 6, both effects are shown. The bottom row frames show the condensation effect. The simulation starts with vapor particles in the upper side of the container. As they meet a very cold surface at the top of the box, vapor particles gradually transition to liquid and fall back.

5.2. Performance analysis

Table 1 show the performance of our approach in seven different scenarios, summarizing the computational impact for each functionality. *Total* shows the total average time per simulation step. For this performance test, we computed 8 PBD internal iterations per step. The table shows that constraint projection takes longer than all other functions. As a reminder, constraint projection is a fundamental step in PBD. An interesting point here is the relatively low impact of our methods, the heat-transfer model and the constraint manager, on the overall simulation. Heat transfer computation takes less than 8.8% of the total time in the worst-case scenario (**Heat transfer** row). Excluding that scenario, the mean value is even lower: 5.64%. Besides, our *constraint manager* takes even less time than the heat transfer. In most cases, the execution time was under 1% of the total cost of a timestep. These results indicate that the additional computational cost imposed by our methods is barely noticeable in relation to the original framework without phase changes at all. It is important to highlight that our method was implemented on the basis of Jan Bender's sequential code [23], available on his official website[2].

5.3. Particle number analysis

We analyzed how the computation time grows with the number of particles. Fig. 8 compares the average computation time per step of simulation for each functionality. For this test, we took the melting demo and varied the number of particles in the bunny model. The chart displays a linear behavior for times in relation to particles number. Visibly, constraint projection takes most of the total time of each timestep. Heat transfer, constraint manager and neighborhood times search times also grow linear with the number of particles, but their growth factor is much lower. This indicates that the percentage of phase-shifting cost in relation to the total cost tends to decrease for larger numbers of particles. In Fig. 9 we compare the computation time among simulations when they have the same number of particles. At a very low number of particles, the neighborhood search is more expensive than the constraint projection (for Vaporization and Condensation). On the other hand, among the other scenarios and particle configurations, the constraint projection takes more time of the total timestep. Despite that, even with larger number of particles the neighborhood search time remains stable and the constraint manager has no noticeable influence in vaporization and condensation effects. The hardest simulation was melting, which took more time independently of the particle number. This was the scenario where our

[2] http://www.interactive-graphics.de/

Table 1
Average times per frame in milliseconds for each scenario.

Scenario	Particles	Total	Neighborhood search	Constraint projection	Viscosity	Heat transfer	Constraint manager
Heat transfer	18k	59.43	6.38	45.047	3.82	3.68	3.66×10^{-2}
Vaporization	18k	60.65	8.64	44.50	3.31	3.64	3.79×10^{-2}
Condensation	20.4k	64.35	15.46	41.33	3.51	3.46	3.96×10^{-2}
Melting	34.6k	409.66	20.37	353.01	6.10	6.32	22.89
Solidification	33.5k	218.46	16.66	179.32	4.67	5.23	11.66
Full Simulation	27.1k	97.39	14.35	71.71	4.35	5.13	1.06

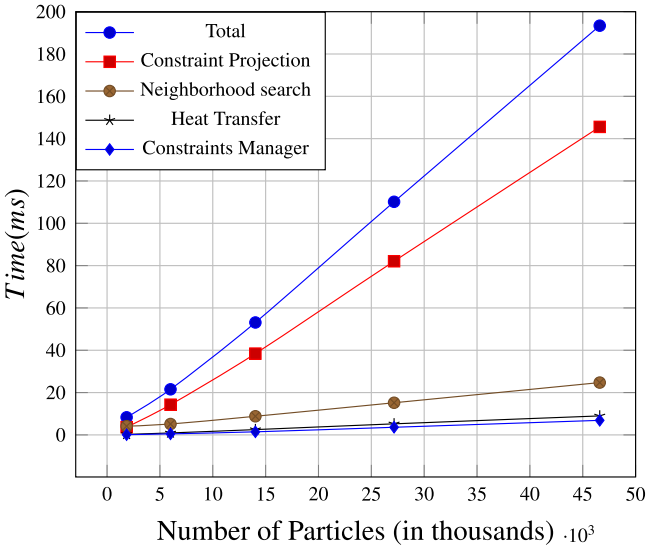

Fig. 8. Computational cost of each functionality according to the number of particles in the model. Notice that the cost for the Heat Transfer and the Constraint Manager are fairly lower than the Constraint Projection (PBD solver) and also grow much slower when the number of particles increase.

constraint manager had the worst behavior. Nevertheless, the constraint manager has lower impact on the whole simulation, even at a large number of particles.

5.4. Parameters and method stability

Table 2 shows the principal variables used along the simulations. As most of them were mentioned on the body of this document, in this section we will just address these parameters regarding the methods stability and values we selected for the simulations.

First, the final value of the distance constraint relies on Young's Modulus E_0 and Linear coefficient of expansion e (listed on Section 3.1). The value of E_0 must be within the 0–1 range because the temperature based E uses this value and an inverse sigma function so that it acts like a damper. Larger values of E_0 lead to instabilities in the solid phase from the first timestep. The main difference between E_0 and α from XPBD is the variation regarding the current temperature of the particles. Likewise, the parameter e is constrained to the range from 0.0 to 0.01. When set to 0, there is no dilation generated from temperature, and the bigger this value the faster a body dilates and contracts. Values above 0.01 cause instabilities because of the velocity of contraction and dilation.

The next set of parameters affects the freezing behavior. nC_{max} controls the maximum number of constraints that one can attach. Despite any value above 0 could be set, the number of constraints highly affects the consistency of solids. Particles with less than 4 distance constraints are quite deformable. Particles with more than 10 constraints turn the simulation unstable because our constraint manager does not create distance constraint with a predefined structure, so particles within the maximum radius for solidification rs_{max} and meeting the solidification temperature will be connected instantaneously with distance constraints. Moreover, rs_{max} values under $2r$ will not create any new constraint due to the minimum distance between particles. Values above $4r$ are suitable to simulate but the behavior of the new solid is somewhat unpredictable: distance constraints very different within the same body causes inconsistency when the body is contracting or dilating.

The third set of parameters in Table 2 are regarding the thermal properties of the material. Those values mainly affect the time for the heat to move from a particle to another. Heat capacity (c) directly influences the amount of latent heat in a given particle (Eq. (21)). So, this value changes the velocity of phase change. We used a constant value of 1000 in our simulations but changing this value will not affect the simulation stability. Values for $T_{melting}$, $T_{evaporation}$, L_{m-f} and L_{e-c} were constant as well, and we used values corresponding to water. Since in real world $T_{melting} < T_{evaporation}$, to meet this inequality is the only restriction to set

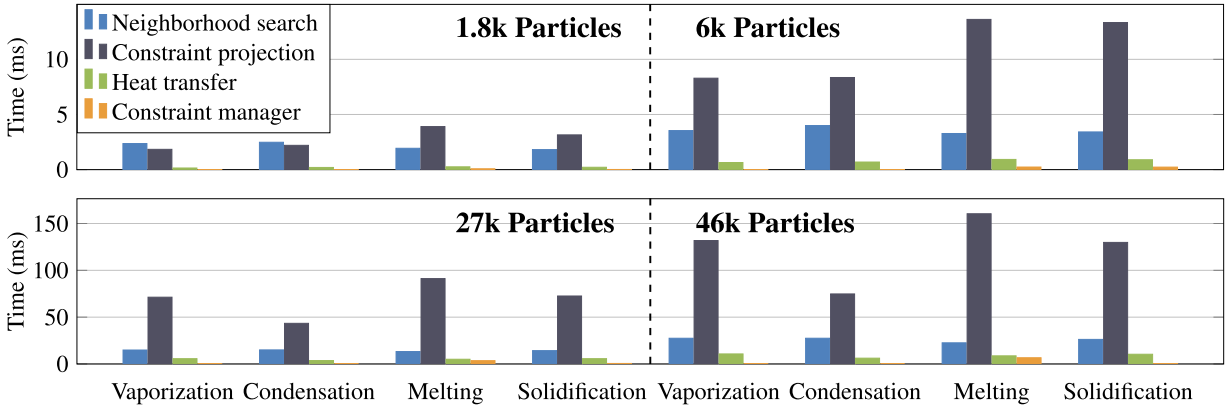

Fig. 9. Performance averages at constant number of particles. Four scenarios are evaluated among four different particle number.

Table 2
Parameters.

Notation	Description	Range	Value
E_0	Initial Young's modulus	(0–1]	1
e	Linear coefficient of expansion	[0.0–0.01]	0
nC_{max}	Maximum constraint number	4–10	4
rs_{max}	Maximum radius for solidification	[2r–4r]	0.055
h	SPH Support radius	–	4r
c	heat capacity	–	1000
$T_{melting}$	Melting Temperature	–	0
$T_{evaporation}$	Evaporation Temperature	–	100
L_{m-f}	Latent Heat melting - fusion	–	333
L_{e-c}	Latent Heat evaporation - condensation	–	2257
k	Thermal conductivity	–	Depends on the demo
ϕ	Diffusion of particle	[10–270]	Depends on the demo
μ	Viscosity	0.025–1	0.025
r	Particle radius	0.01–0.025	0.025
ρ_{solid}	Density of solid particle	[700–1000]	900
ρ_{fluid}	Density of fluid particle	[700–1000]	1000
ρ_{gas}	Density of gas particle	[700–1000]	700
m	Mass of particle	[0.0064–0.07]	$0.8{*}8r^{3}{*}\rho_{solid}$
α	Compliance	[0–1]	$0.16e-9$

those values. An incorrect selection of threshold temperatures will lead to artifacts because particles with distance constraints will arise, compromising the stability of the model. Moreover, L_{m-f} and L_{e-c} together with c define the time that a phase change takes in the simulation. Again, an incorrect value selection will not affect the stability but it is recommended to use real world values to obtain a simulation within a reasonable phase change time. Thermal conductivity (k) and ϕ are closely related in the sense that those variables define the heat transfer velocity on each of the schemes: Explicit or Cleary and Monaghan. Selecting the correct values for either of the models should cause the same behavior, but the Cleary model advantage is the capability to work with different materials interactions. The final value depends on the k value of each neighbor particle, while in the explicit model, ϕ is the same for each neighbor particle. Thus, it is not the selection of heat model but the parameter of each model that affects the behavior of the simulation.

5.5. Discussion and limitations

In this sequential implementation of PBD, simulations up to 8k particles are interactive. Above this number, the framerate drops to non-interactive frequencies. In comparison with the basic PBD code, our approach increases the computational cost by only 8% in the worst case. In current optimized parallel implementations of PBD, such as that of the Flex library, models containing near 100k particles can reach interactive framerates. This indicates that an eventual integration of our algorithms with a GPU implementation of PBD, e.g. using CUDA, may increase performance by at least one order of magnitude.

In all our tests we ignored heat loss due to particles interaction with the atmosphere. This leads to inaccurate behavior in some situations. For instance, when gas particles are raising from a boiling fluid, they should lose their energy when in contact with air (or another gas) particles in the atmosphere. Moreover, as we simply use distance constraints to simulate solids, we do not achieve rigid body behaviors. Instead, our solid bodies allow deformation. This could be improved by introducing more consistent constraints to the constraint manager. A cluster approach to model rigid elements, as available in the Flex library, should also be investigated further. As our constraint manager creates distance constraints when particles meet the two requirements (Temperature of both particles and distance between particles), the newly formed solids are not totally consistent. It is necessary to improve the constraint manager to create structured constraints depending on the information of the current constraints in the particle.

6. Conclusions

We presented a full Lagrangian method to simulate bidirectional phase shifting materials as a function of temperature. Solid, liquid and gas states were modeled with position based dynamics (PBD) as well as the phase-shifting processes of melting, solidification, vaporization, and condensation. Modifications to PBD density-, viscosity- and distance-constraints were introduced for the first time to simulate the thermal phenomena.

A latent heat model was also implemented to drive the transitions between the different states. A heat transfer model was adapted to comply with the law of thermodynamics that defines the heat transfer flow direction.

More complex phenomena such as sublimation, deposition or even plasma state were left out because they are less common. However, they are worth to be explored in future works. Another future development could explore the viscosity method proposed by Takahashi et al. [30] that could lead to soft melting/fusion transitions.

The core functions we proposed to simulate phase changes took, in the worst case, only 8% of the total timestep duration. This makes us believe that a parallelized version of PBD extended with our approach could equally be interactive with a considerably larger number of particles.

As our work is temperature-based, we need to obtain latent heat from temperature change. Introducing a fully energy-based model, although more computationally intensive, would be a more physics-based way to represent phase change and thermal phenomena in general. The proposed model, nevertheless, covers heat transfer by contact and, in our implementation, imitates convection. Future implementations must also include heat transfer by radiation, but none of these limitations impact the PBD constraints here proposed.

Better melting effects could be obtained using the very recent method by Weiler et al. [8]. However, their method carries more complex calculations.

Finally, the gas simulation could be improved developing new methods in SPH or PBF capable of handling smoke particles without the need for filling all the domain with atmospheric particles, since it is computationally impractical. In this topic, the work of Ren et al. [11] is worth to be explored.

Acknowledgments

We thank to Professor Jan Bender for making public his PBD source code and Iago Berndt for his support in the flex-based rendering. This study was partly funded by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001 and partially by Petrobras (Project Annelida).

We also acknowledge FAPERGS (project 17/2551-0001192-9) and CNPq-Brazil (project 311353/2017-7) for their financial support.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.cag.2018.09.004.

References

[1] Nealen A, Müller M, Keiser R, Boxerman E, Carlson M. Physically based deformable models in computer graphics. Comput Graph Forum 2006;25(4):809–36.
[2] Gao Y, Li S, Yang L, Qin H, Hao A. An efficient heat-based model for solid-liquid-gas phase transition and dynamic interaction. Graph Models 2017a;94:14–24.
[3] Stomakhin A, Schroeder C, Jiang C, Chai L, Teran J, Selle A. Augmented mpm for phase-change and varied materials. ACM Trans Graph TOG 2014;33(4):138.
[4] Hochstetter H, Kolb A. Evaporation and condensation of sph-based fluids. In: Proceedings of the ACM SIGGRAPH/eurographics symposium on computer animation. ACM; 2017. p. 3.
[5] Farrokhpanah A, Bussmann M, Mostaghimi J. New smoothed particle hydrodynamics (SPH) formulation for modeling heat conduction with solidification and melting. Numer Heat Transf Part B Fundam 2017;71:299–312.
[6] Müller M, Heidelberger B, Hennix M, Ratcliff J. Position based dynamics. J Vis Commun Image Represent 2007;18(2):109–18.
[7] Gao Y, Li S, Qin H, Hao A. A novel fluid-solid coupling framework integrating flip and shape matching methods. In: Proceedings of the Computer Graphics International Conference. ACM; 2017b. p. 11.
[8] Weiler M, Koschier D, Brand M, Bender J. A physically consistent implicit viscosity solver for SPH fluids. Comput Graph Forum 2018;37(2):145–55.
[9] Müller M, Solenthaler B, Keiser R, Gross M. Particle-based fluid-fluid interaction. In: Proceedings of the 2005 ACM SIGGRAPH/eurographics symposium on computer animation. ACM; 2005. p. 237–44.
[10] Cleary PW, Monaghan JJ. Conduction modelling using smoothed particle hydrodynamics. J Comput Phys 1999;148(1):227–64.
[11] Ren B, Yan X, Yang T, Li C-f, Lin MC, Hu S-m. Fast SPH simulation for gaseous fluids. Vis Comput 2016;32(4):523–34.
[12] Ihmsen M, Orthmann J, Solenthaler B, Kolb A, Teschner M. SPH fluids in computer graphics. In: Lefebvre S, Spagnuolo M, editors. Eurographics 2014 - state of the art reports. The Eurographics Association.
[13] Solenthaler B, Pajarola R. Predictive-corrective incompressible SPH. In: Proceedings of the ACM transactions on graphics (TOG), 28. ACM; 2009. p. 40.
[14] Ihmsen M, Cornelis J, Solenthaler B, Horvath C, Teschner M. Implicit incompressible sph. IEEE Trans Vis Comput Graphics 2014b;20(3):426–35.
[15] Becker M, Teschner M. Weakly compressible sph for free surface flows. In: Proceedings of the 2007 ACM SIGGRAPH/eurographics symposium on computer animation. Eurographics Association; 2007. p. 209–17.
[16] Macklin M, Müller M. Position based fluids. ACM Trans Graph 2013;32(4):104.
[17] Keiser R, Adams B, Gasser D, Bazzi P, Dutré P, Gross M. A unified lagrangian approach to solid-fluid animation. In: Proceedings of the symposium on point-based graphics, eurographics/IEEE VGTC. IEEE; 2005. p. 125–48.
[18] Gray J, Monaghan J, Swift R. Sph elastic dynamics. Comput Methods Appl Mech Eng 2001;190(49–50):6641–62.
[19] Becker M, Ihmsen M, Teschner M. Corotated SPH for deformable solids.. In: NPH. Citeseer; 2009. p. 27–34.
[20] Macklin M, Müller M, Chentanez N, Kim T-Y. Unified particle physics for real–time applications. ACM Trans Graph 2014;33(4):1–12.
[21] Macklin M, Müller M, Chentanez N. Xpbd: position-based simulation of compliant constrained dynamics. In: Proceedings of the 9th international conference on motion in games. ACM; 2016. p. 49–54.
[22] McNaught AD, McNaught AD. Compendium of chemical terminology, 1669. Blackwell Science Oxford; 1997.
[23] Bender J, Müller M, Macklin M. Position-based simulation methods in computer graphics. In: EUROGRAPHICS 2015 tutorials. Eurographics Association; 2015.
[24] Müller M, Charypar D, Gross M. Particle-based fluid simulation for interactive applications. In: Proceedings of the 2003 ACM SIGGRAPH/eurographics symposium on computer animation. Eurographics Association; 2003. p. 154–9.
[25] Schechter H, Bridson R. Ghost SPH for animating water. ACM Trans Graph 2012;31(4):1–8.
[26] Shao S, Lo EY. Incompressible SPH method for simulating newtonian and non-newtonian flows with a free surface. Adv Water Resour 2003;26(7):787–800.
[27] Hu X, Adams NA. An incompressible multi-phase SPH method. J Comput Phys 2007;227(1):264–78.
[28] Russell S. Open dynamics engine v0. 5 user guide. Comput Graph 2007;176(2):121–36.
[29] Akinci N, Ihmsen M, Akinci G, Solenthaler B, Teschner M. Versatile rigid-fluid coupling for incompressible SPH. ACM Trans Graph 2012;31(4):1–8.
[30] Takahashi T, Nishita T, Fujishiro I. Fast simulation of viscous fluids with elasticity and thermal conductivity using position-based dynamics. Comput Graph Pergamon 2014;43(1):21–30.