Contents lists available at ScienceDirect

Computers & Graphics

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

Technical Section

A heuristic approach to the simulation of water drops and flows on glass panes

Kai-Chun Chen, Pei-Shan Chen, Sai-Keung Wong*

National Chiao Tung University, Taiwan

A R T I C L E I N F O

Article history:
Received 23 March 2013
Received in revised form
9 August 2013
Accepted 28 August 2013
Available online 7 September 2013

Keywords:
Water drops
Water flows
Simulation
Particle system
Height map

A B S T R A C T

Water drops and water flows exhibit interesting motion behavior. In this paper, we adopt a simple but effective approach for simulating this behavior on glass panes in a physically plausible manner. We combine a particle system and a height map to compute their movements and shapes. Our approach efficiently handles the merging of water drops and the formation of residual water droplets. We report our results for several examples of water behavior simulated in real time. The experimental results show that our system simulates water drops and water flows with high quality.

© 2013 Elsevier Ltd. All rights reserved.

1. Introduction

Water drops and water flows exhibit interesting motion behavior and form amazing patterns on the surfaces of objects, such as the leaves of plants and glass panes. These drops and flows contain a small amount of water and are commonly observed on a rainy day. Their motion is affected by various factors, including gravity, surface tension, the roughness of the surface to which they adhere, surface purities, cohesion and adhesion. Cohesion (adhesion) is the tendency of similar or identical (dissimilar) particles to attract each other. Simulating the motion behavior of water drops and flows is a challenging problem if all of these factors are considered. Alternatively, we simulate them based on several simple heuristic rules.

Figs. 1 and 2 show a series of images of water drops and water flows on a glass pane. These images were captured in real life. On a rainy day, rainwater hits a glass pane, with some rainwater forming water drops. These water drops merge if they touch each other. Due to cohesion and gravity, most of the water moves to the lower part of the new water drop in a short time. Running water drops become water flows. Water flows meander down a glass pane instead of running directly downward. Fig. 2 shows residual water drops that are left behind by a water flow. These residual drops are formed because the surface of the material to which the drops attach themselves has strong adhesion toward water, among other factors. A small amount of water forms the residual water droplets after a water flow passes. The process of producing residual water droplets is continuous and smooth. Furthermore, it can be observed that the direction of movement of a water flow mainly depends on that of the front of the water flow.

There are a variety of particle-based techniques that have been proposed for simulating water drops and water flows. However, most of them do not address the formation of residual water drops, which is one of the important features to consider in simulating water flows. In this paper, we combine a particle system with a height map to simulate three different behaviors of water drops and water flows: (1) the merging of water drops, (2) the meandering of water flows and (3) the formation of residual water drops. The movements and shapes of water flows are computed based on simple rules so that the simulated results can be computed at real-time rates. There are three major advantages of our method over the existing particle-based techniques:

1. An ID map is adopted to efficiently determine whether to merge water drops and water flows. This technique works well even though the shapes of water drops and water flows are irregular.
2. Our method applies smoothing and erosion operators so that the shapes of water drops and water flows change in a continuous manner.
3. Our method simulates the formation of residual water droplets realistically.

* Corresponding author. Tel.: +886 3 5712121x56626.
E-mail address: cswingo@cs.nctu.edu.tw (S.-K. Wong).

0097-8493/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cag.2013.08.004

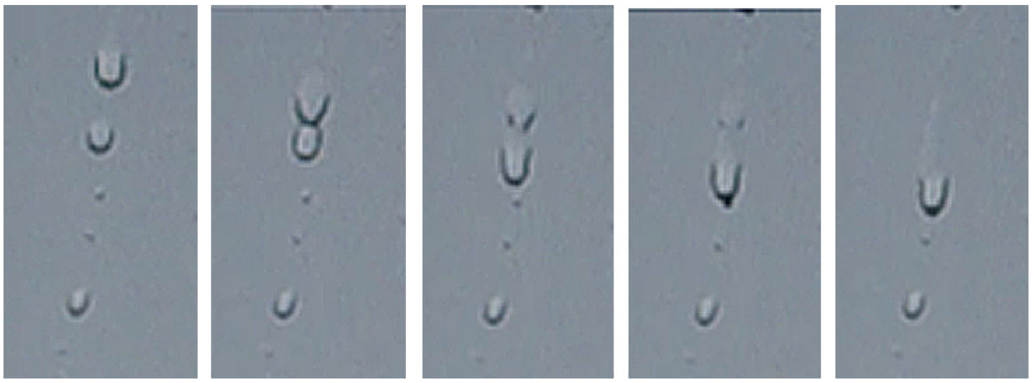

Fig. 1. A series of snapshots captured in real life. Two water drops make contact and then merge into a new water drop.

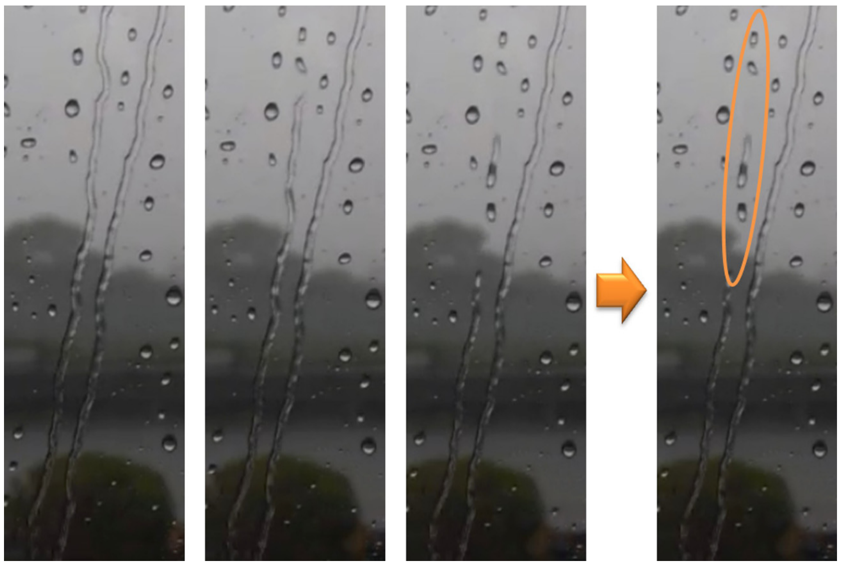

Fig. 2. Residual droplets (upper middle part) left behind by a water flow. The residual droplets are highlighted in the right image.

2. Related work

In this section, we review the techniques that are related to the technique proposed in our work. In Kaneda et al. [6], a glass plate is divided into a grid and a water droplet is represented as a particle. The movement of water droplets is affected by various factors, such as the surfaces affinity for water and the masses of droplets. The law of conservation of momentum is applied for merging droplets. A ray-tracing technique is adopted to render water droplets, represented as spheres.

In Fournier et al. [3], water drops are animated on triangular meshes. A water drop is modeled as a deformable hemisphere that is simulated by a mass–spring system. The movement of water drops is affected by factors such as friction, adhesion, roughness and collisions between water drops. Streaks are added along the paths of running water drops, and the roughness of the surface is altered along the paths. This method involves collision detection between water drops at discrete frames. However, some collision events may remain undetected, resulting in the unnatural movement of water drops.

Kaneda et al. [7] proposed an approach for animating water droplets on a windshield for driving simulators. In the study, the simulation space was discretized into a set of quadrilateral meshes using a method similar to that proposed in Kaneda et al. [8]. An iterative method was proposed to handle the interaction between water drops and obstacles, such as wipers. Four factors were considered in computing the motion of water drops: (1) Newton's law of motion; (2) the affinity for water of the neighboring regions; (3) whether the neighboring regions were wet or dry; and (4) the existence of obstacles. A probabilistic model was also proposed for computing the direction of movement of a water drop. In general, the greater the speed of a droplet, the higher the probability the droplet moves in the same direction.

Jonsson and Hast [5] employed bump maps to simulate the flow of a droplet on structured surfaces. The normal vectors of a bump map capture the geometry of a micro-structured surface. The method proposed by Jonsson and Hast can be used to simulate a droplet that leaves a trail. This is achieved by changing the glossiness attribute of the texels of a surface texture. In Takenaka et al. [12], a moving water drop is represented as multiple hemispheres (the head and tail of the water drop). The larger the mass of a moving water drop is, the longer its shape becomes. There is no shrinking effect for water flows. The contours of water drops are checked for collision between drops. However, after two water drops merge into a new water drop, only one hemisphere is immediately displayed. Thus, the merging process of water drops is not natural.

Sato et al. [10] employed graphics hardware to simulate water droplets. Water droplet textures were created by rendering water droplets as polygonized hemispheres. The depth of field effect was applied to enhance the visual quality of the drops. The results clearly show individual hemispheres that are unnatural. Stuppacher and Supan [11] simulated the motion of water drops on a 2D texture. The amount of water flowing downward was determined by a remainder map. No specific method for merging water drops was clearly stated. To render the motion of water drops, a texture was created based on a height map in which each texel recorded the amount of water. The height map was then converted to a normal map. Motion was rendered with lighting based on a Fresnel model. Although the height map was smoothed before creating the normal map, the method of Stuppacher and Supan does not yield a smooth shape for running water droplets. In some cases, the water flows are discontinuous. Furthermore, because a water flow can move only from one texel to the adjacent texel, their method may not properly handle fast running water drops and water flows. Yvonne and Johannes [17] proposed a GPU-based method to simulate droplet flows on 3D surfaces, which are decomposed into charts. Their method handles droplet flows that leave a trail. All fluid information is stored in texture memory. Each texel maps to a certain part of the surface, and the texel stores the velocity and the amount of water of the corresponding fluid particle. Tatarchuk [13] proposed a variety of techniques for rendering rainy-day effects using GPUs.

Yu et al. [16] employed metaballs to model the shape of water drops under the effects of a gravitational field. Their method constructs the surface of a droplet with an irregular boundary. In Algan et al. [1] the direction of movement of a water drop depends on the amount of water in front of the water drop. To render a water drop, a set of metaballs is used. A large blob and a smaller, flatter blob are placed in a lower and upper section, respectively. The method does not handle the residual water droplets that are left behind by water flows. Nakata et al. [9] simulated water drops formed on a hydrophobic windshield. The required parameters, such as the acceleration of water drops versus wind velocity and the distribution of the number of raindrops for each diameter, were obtained from experiments (see [4]).

Wang et al. [14] developed a height field-based system and simulated a variety of water effects, such as shallow waves, water drops, rivulets, capillary events and fluid/floating rigid body coupling. Water motion was computed based on the general shallow wave equations. Level-set approaches have also been proposed for simulating water drops and water flows. Wang et al. [15] proposed a physics-based method that sets the contact angles at the intersection of fluid-free surfaces and solid objects. A virtual surface method was proposed to modify the level-set distance field to compute the contact angle accurately. A sparse grid representation was employed for simulation. However, the grid must be refined to model the surface details of water drops. Zhang et al. [18] employed an implicit mean curvature flow operator to model surface tension effects. The droplet shapes were constructed so as to maintain the contact angle between water drops and solid surfaces. The method of Zhang et al. can be used to model the shape of water drops as meshes. Hence, mesh connectivity must be updated to handle topological changes.

3. System overview

Fig. 4 shows our system pipeline. Our system has two layers: a physical simulation layer and a rendering layer. The physical simulation layer consists of a particle system and a height map. In the particle system, a particle represents a water drop or the front of a water flow. The trail of a moving water drop is modeled by updating the height map in each time step. We assume that water drops and water flows move over a surface that is discretized uniformly into an $N \times M$ grid, as shown in Fig. 3. The length of a side of a grid cell is ℓ. The value of the affinity for water is inside the interval $[0, 1]$ and is randomly computed initially for each grid cell. The height map has two purposes: (1) recording the amount of water and (2) constructing the shape of water drops and water flows.

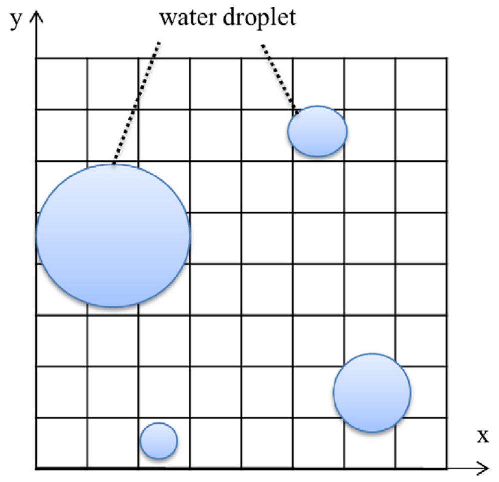

Fig. 3. The surface is discretized into a set of uniform grid cells.

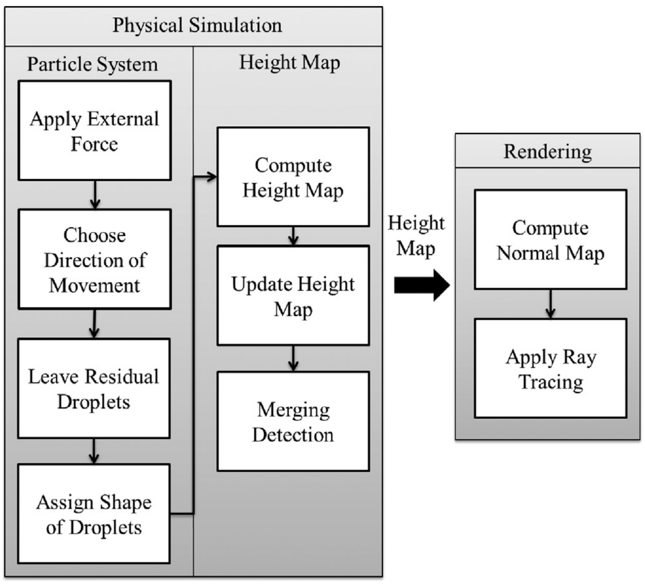

Fig. 4. The system pipeline. In each simulation time step, the simulation portion is performed and followed by the rendering portion.

The physical simulation layer performs the following tasks at each time step:

- *Apply external force*: Compute external forces such as gravity and friction.
- *Choose direction of movement*: Compute the directions of movement of the water drops and water flows.
- *Leave residual droplets*: Determine whether residual droplets are created for a water flow.
- *Assign shape of droplets*: Construct the shapes of water drops by using one or multiple hemispheres.
- *Compute height map*: Construct a height map.
- *Update height map*: Update the height map by applying a smoothing operation and an erosion operation.
- *Merging detection*: Merge the water drops and water flows based on an ID map.

In the rendering layer, a normal map is computed based on the height map. Then, an environment map and Fresnel term are employed to render the final simulated results.

4. Particle system

The size of the simulation time step is denoted as Δt. A particle i has several attributes, e.g., position $\mathbf{p}_i$, mass m_i and velocity $\mathbf{v}_i$. $\mathbf{p}_i$ is computed as $\mathbf{p}_i = \mathbf{p}_i^0 + \mathbf{v}_i \Delta t$, where $\mathbf{p}_i^0$ is the position of particle i in the previous time step. A particle i has a unique ID (*id*entifier) which is denoted as ID_i. A static water drop starts to move if its mass is greater than a predefined threshold m_{static}, which is a user-defined value. In each time step, we generate the particles randomly on the surface. Approximately 80% of the newly created particles have masses that are less than m_{static}.

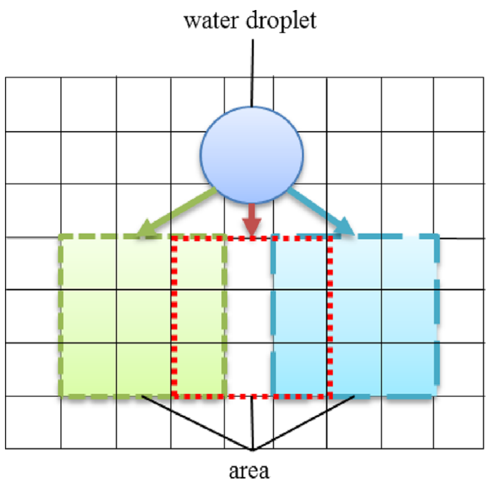

Fig. 5. The three regions used to compute the amount of water in front of a water droplet.

4.1. Computation of particle velocity

The movement of a water drop depends on various factors that make the water drop meander downward. Instead of considering all of the factors, we adopt a simple approach that simplifies the computation. We compute the external forces exerted on particle i, including gravity $m_i\mathbf{g}$ and friction $\mathbf{f}_i$, to estimate the speed s_i of i, where $\mathbf{g}$ is the gravitational constant. The net external force exerted on i is equal to $(m_i\mathbf{g}-\mathbf{f}_i)$. We let $\tilde{\mathbf{a}}_i = (m_i\mathbf{g}-\mathbf{f}_i)/m_i$ and $\tilde{\mathbf{v}}_i = \mathbf{v}_i^0 + \tilde{\mathbf{a}}_i \Delta t$, where the superscript 0 denotes the state in the previous time step. Thus, $s_i = \|\tilde{\mathbf{v}}_i\|$. In our experiment, $\mathbf{f}_i = m_{\text{static}}\mathbf{g}$. After determining the speed of the particle, we compute the direction of movement and velocity of the particle.

We consider the cohesive force and the affinity of a surface for water to determine the direction of movement $\mathbf{w}_i$ of particle i, where $\mathbf{w}_i$ is a unit vector. We check the grid cells in front of the particle. If there is water in front of the particle, the particle moves to the water and merges with it. In our protocol, we verify the amounts of water in three regions in front of the particle, as shown in Fig. 5. A region consists of a 3×3 cell grid. The particle moves to the region with the most amount of water. If there is no water in front of the particle, the particle moves to the region with the highest affinity for water. The higher the affinity for water is, the

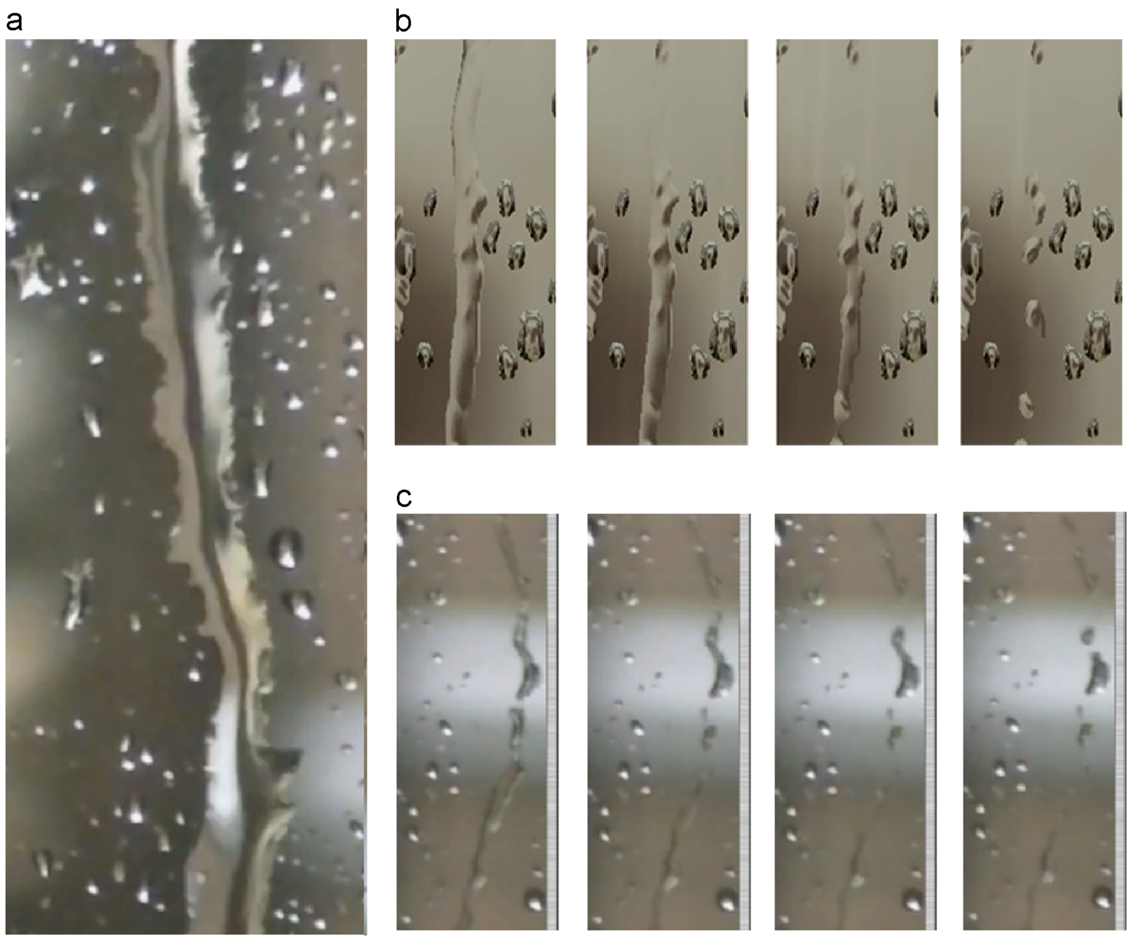

Fig. 6. (a) A real-world water flow on a glass pane. (b) The water flow and residual water droplets simulated in our system. (c) A water flow and residual water droplets in the real world. Our simulated result is similar to the example captured in the real world.

easier water can move. However, if there is no difference in the affinity among the regions, we randomly compute a direction pointing downward. After the direction of movement $\mathbf{w}_i$ is determined, the velocity $\mathbf{v}_i$ of particle i is set to $s_i\mathbf{w}_i$.

4.2. Computation of residual droplets

As a water flow moves, some amount of water contained in the water flow is left behind and several residual water droplets may be formed along the flows trail. Algorithm 1 shows the steps followed to generate these residual water drops. An attribute τ_i of particle i indicates the elapsed time over which no residual water drop has been created. Let $\tau_{\max}$ be a user-defined value for controlling the maximum period of generating a residual water droplet. $\tau_{\max}$ should be greater than Δt. A probability function $\mathrm{pr}(\tau_i, \Delta t, \tau_{\max})$ is defined to compute the probability that a residual water drop is created at the current time step. In our implementation, we have

$$\mathrm{pr}(\tau_i, \Delta t, \tau_{\max}) = \min\left(1, \beta\frac{\Delta t}{\tau_{\max}}\min\left(1, \frac{\tau_i}{\tau_{\max}}\right)\right), \tag{1}$$

where β is a constant. Consider a time interval $[0, t_1]$ for $t_1 \in [0, \tau_{\max}]$ and the simulation time step tends to zero. Thus, the expected elapsed time for generating a residual water droplet is equal to $\int_0^{t_1} t(\beta t/\tau_{\max}^2)\,dt = \beta t_1^3/3\tau_{\max}^2$. For $t_1 = \tau_{\max}$, we set $\beta = 3$ and Δt should be less than or equal to $\tau_{\max}/3$. When a residual water droplet is generated, τ_i is reset to zero. Therefore, a water flow may leave several water droplets along its path. Notice that the newly created residual water droplet is static, and thus, its mass is less than or equal to m_{static}. As shown in Fig. 6, our simulated results are similar to the corresponding real-world example. In our formulation, the probability of producing a residual water droplet increases as the water drop or water flow moves over a greater distance. Other probability functions can also be used.

Algorithm 1. Residual droplet, $\tau_{\max} \leftarrow 0.4$.

```
for each moving particle $i$ do
  /* LeaveResidual determines whether or not to generate
  * a residual water droplet with probability $\mathrm{pr}(\tau_i, \Delta t, \tau_{\max})$*/
  if LeaveResidual($\mathrm{pr}(\tau_i, \Delta t, \tau_{max})$)==true then
     create a new particle $i'$ ;// $i'$ represents a residual water
    droplet
    $\mathbf{p}_{i'} \leftarrow \mathbf{p}_i$; // position
    $\alpha \leftarrow$random$(0.1, 0.3)$ // a random number between [0.1, 0.3]
    $m_{i'} \leftarrow \min\{m_{\text{static}}, \alpha m_i\}$
    $m_i \leftarrow m_i - m_{i'}$
    $\mathbf{v}_{i'} \leftarrow \mathbf{0}$
    $\tau_{i'} \leftarrow 0$
    $\tau_i \leftarrow 0$
  else
    $\tau_i \leftarrow \tau_i + \Delta t$
  end if
end for
```

5. Height map

The height map is denoted as H and over a grid cell $g(x,y)$ at position (x,y). The height value $H(x,y)$ of the grid cell $g(x,y)$ indicates the amount of water at $g(x,y)$. We use an ID map to record the region occupied by a water drop or water flow. An ID map I has the same resolution as the height map. Each grid cell $g(x,y)$ of the ID map stores the ID $I(x,y)$ of a water drop or the ID of a water flow. If a grid cell $g(x,y)$ is not covered by any water drop or water flow, then the value of $I(x,y)$ is set to -1. Initially, the height value and ID of each grid cell are set to zero and -1, respectively.

For a moving particle i, we place a hemisphere at $\mathbf{p}_i$. If the moving particle represents a water flow, the trail of the water flow is handled by updating the height map. The shape of static water drops is modeled by multiple hemispheres, as will be detailed in Section 5.3.

5.1. Height map construction

In each simulation step, we update the height values of the grid cells that are affected by the particles. The height values of the other grid cells are not changed in this step. In the following, we describe how we update the height values of the grid cells. For each moving particle i, we set a hemisphere with radius $r_i = \sqrt[3]{3m_i/2\rho\pi}$ at position $\mathbf{p}_i$, where ρ is the density of water. The height values of the grid cells covered by the hemisphere are computed. Consider a grid cell at position $\mathbf{p} = (x,y)$. The height value $H(x,y)$ is set to $\sqrt{r_i^2 - (\mathbf{p}-\mathbf{p}_i)^2}$ if the following two conditions are satisfied:

1. $r_i^2 - (\mathbf{p}-\mathbf{p}_i)^2 > 0$.
2. The current height value at (x, y) is smaller than $\sqrt{r_i^2 - (\mathbf{p}-\mathbf{p}_i)^2}$.

If one of the conditions is not satisfied, the height value of the grid cell is not changed. As shown in Fig. 7, the height values of the grid cells are higher if the grid cells are near the center of the hemisphere. At the same time, $I(x,y)$ is set to the ID_i if the two conditions are satisfied. Notice that if the previous value of $I(x,y)$ is non-negative

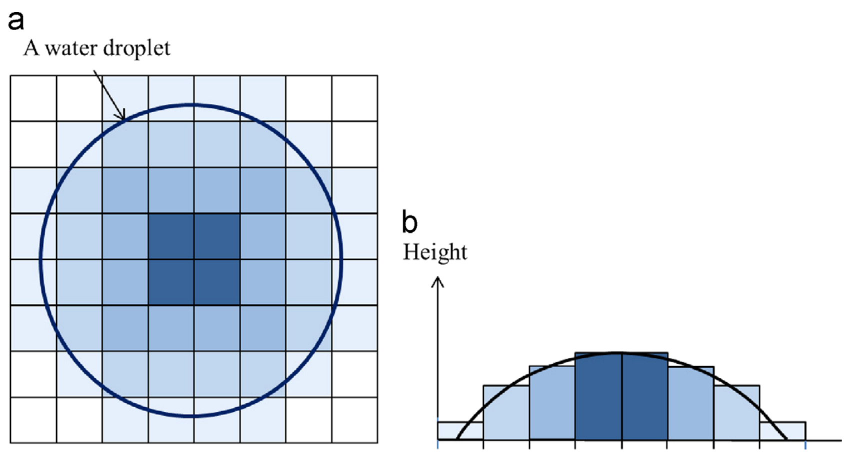

Fig. 7. (a) Top-down view and (b) side view of a water droplet on grid cells. The darker the grid cell is, the higher its height value becomes.

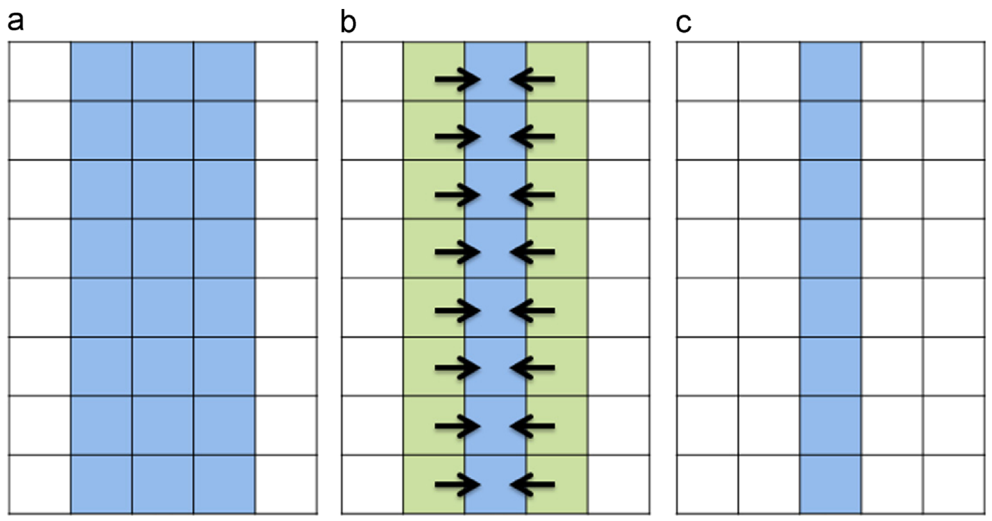

Fig. 8. The erosion operation. (a) Blue cells represent a water flow. (b) Green cells (with an arrow) represent the boundaries of the water flow. The green cells are shifted toward the middle of the water flow. (c) After the erosion operation is performed, the water flow becomes thinner. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

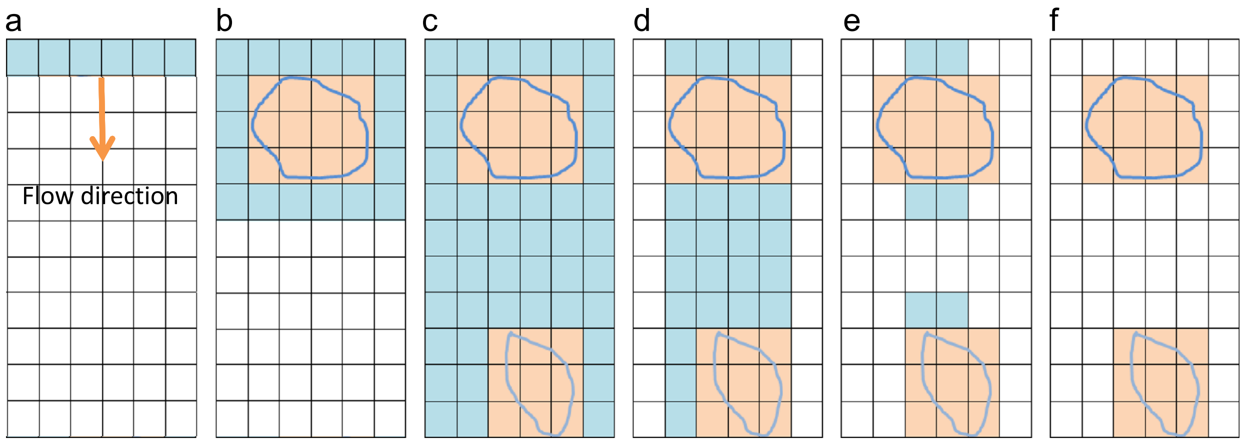

Fig. 9. Application of the smoothing and erosion operations to produce the shapes of residual water droplets. Blue squares represent a water flow; light-salmon squares (regions enclosed with curves) represent residual water droplets. As the water flow moves, we generate a residual water droplet with a certain probability. The residual water droplets appear gradually due to the smoothing and erosion operations applied to the height map. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

and it is not equal to ID_i, then the grid cell is occupied by another water drop or water flow. Then, a merging process is performed for the pertinent water drops and water flows.

5.2. Smoothing and erosion operations

We observe that a water flow shrinks inward as it flows. This behavior occurs because the water moves downward and there is no more water flowing through the path of the water flow. To simulate this phenomenon, we apply two simple operations, smoothing and erosion, which are employed to modify the entire height map after we update the height map according to the positions of particles.

The smoothing operation is described as follows. In each time step, the height value of a grid cell is computed as the average of the height values of the neighboring 3×3 grid cells (including itself). If the height value of a grid cell is less than ε_h, then we set the height value of the grid cell to zero. In our experiment, we set $\varepsilon_h = 0.01$. After performing the smoothing operation, the shapes of water flows become flatter.

The erosion operation is used to shrink the water flows inward. We detect the boundary of the water flow by checking the height map. Consider a grid cell $g(x,y)$. If $H(x,y) > 0$ and $H(x+1,y) = 0$, then $g(x,y)$ is a boundary grid cell. Fig. 8 illustrates an example in which there are no residual droplets. In this case, we shift the height values of the grid cells inward. If there are residual droplets, we execute the following steps. When we perform the erosion operation, we detect whether the boundary grid cell lies inside the region of a residual droplet. If it does, then we do not change the height value of the grid cell. If a boundary grid cell is not covered by any residual water droplets, we perform the erosion operation as usual. After performing the erosion operation, the water flow shrinks inward and becomes water drops. Fig. 9 illustrates an example involving the formation of residual water droplets. Both the smoothing and erosion operations are performed.

5.3. Shape of static water drops

Fig. 10 shows water drops with different shapes. In general, the shapes of the water drops on a surface with a high affinity (i.e., on a hydrophilic surface) are flatter than those on a surface with a lower affinity (i.e., on a hydrophobic surface). To construct the shape of a static water drop, we adopt an approach similar to that proposed in Chen et al. [2].

We use one or more than one hemisphere to model the shape of a static water drop. By using multiple hemispheres, we can

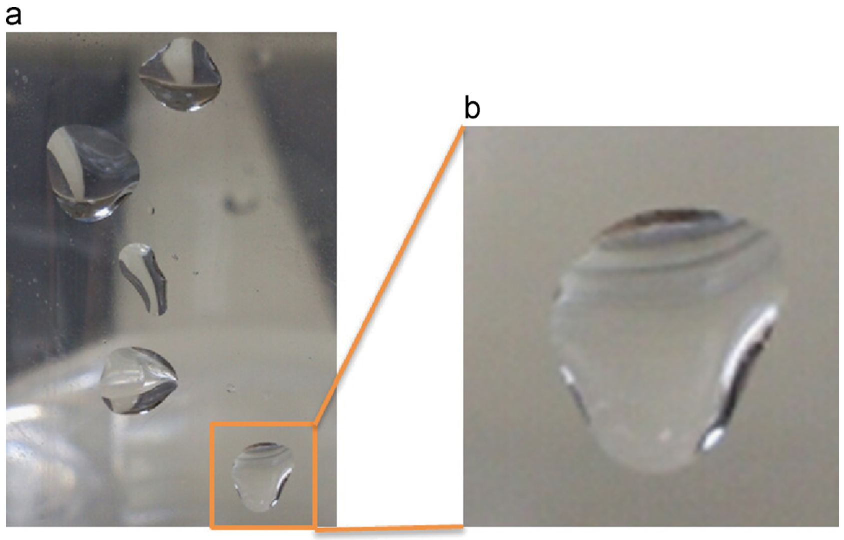

Fig. 10. (a) Real water droplets with irregular shapes on a bottle. (b) A magnification of the water drop highlighted in (a). Due to surface tension, the surface of the water drop is smooth.

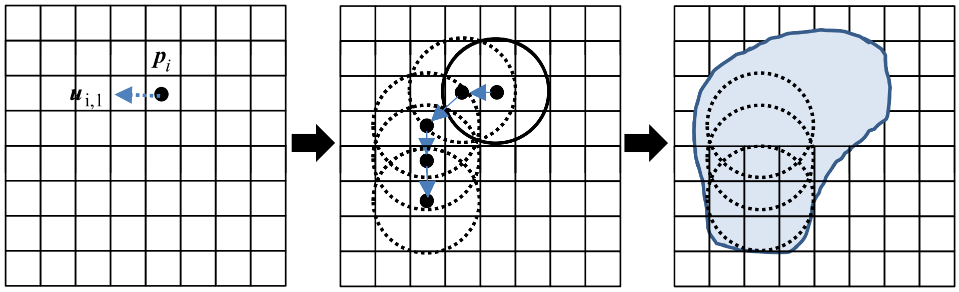

Fig. 11. The process of reconstructing the shape of a static water drop. Left: computation of directions. Middle: computation of the points $\mathbf{q}_{i,j}$ and hemispheres placed at each point. Right: after performing the smoothing operation to the height map, the shape of the static water drop becomes smooth.

model different shapes of static water drops. Fig. 11 shows the steps for constructing the shape of a static water drop.

We compute N_i positions $\{\mathbf{q}_{i,1}, \mathbf{q}_{i,2}, \dots, \mathbf{q}_{i,N_i}\}$, where N_i is a random number within the interval $[1,5]$, and place a hemisphere at each position. The position of the first hemisphere, i.e., $\mathbf{q}_1^i$, is set to $\mathbf{p}_i$. If $N_i = 1$, the step is complete. Otherwise, we adopt the following formula to compute the other positions one by one:

$$\mathbf{q}_{i,j+1} = \mathbf{q}_{i,j} + d_{i,j}\mathbf{u}_{i,j}, \tag{2}$$

where $j \in [1, N_i - 1]$, $\mathbf{u}_{i,j}$ is a unit direction and $d_{i,j}$ is the distance between $\mathbf{q}_{i,j}$ and $\mathbf{q}_{i,j+1}$. Because our simulation space is a grid, the set of direction vectors for $\mathbf{u}_{i,j}$ is $\{(\cos\theta, \sin\theta) | \theta \in \{0, \pi, 5\pi/4, 3\pi/2, 7\pi/4\}\}$. Furthermore, $d_{i,k}$ is set to ℓ or $\sqrt{2}\ell$ depending on the direction. We assume that the mass of the N_i hemispheres is the same. Hence the radius of a hemisphere is computed as $r = \sqrt[3]{(3\pi/4)(2m_i/N_i) \times \rho}$, where ρ is the density of water. In our experiments, ρ is set to one. In the final step, we apply a smoothing operation to the height map to make the shape smooth. The set of positions $\{\mathbf{q}_{i,j} | \forall j \in [1, N_i - 1]\}$ are reused in the next time step if particle i does not move.

5.4. Interpolation

The displacement of a particle between two consecutive time steps is $\|\mathbf{p}_i - \mathbf{p}_i^0\|$. If $\|\mathbf{p}_i - \mathbf{p}_i^0\| > \ell$, then the particle may move by more than one grid cell. We call this type of particle as a *fast running particle*. Water drops modeled as fast running particles jump (teleport) from their current grid cell to the new grid cell in the next time step. To tackle this problem, the intermediate grid cells lying between $\mathbf{p}_i$ and $\mathbf{p}_i^0$ are computed according to the line rasterization algorithm in computer graphics so that the intermediate grid cells are connected.

5.5. Discussion

In each time step, the height map is updated for the particles. The height values of the grid cells covered by the static water drops are retained. On the other hand, the height values of the other grid cells that are covered by the moving particles are gradually reduced due to the smoothing operation. The height values of the boundary grid cells are also reduced due to the erosion operation. Consequently, our method can simulate water flows flowing downward and the trails of water flows, which become thinner over time. We notice that after performing the smoothing operation, some grid cells with zero height are modified, and their height values become non-zero. If we perform the smoothing operation several times, more grid cells will have non-zero height values. This is an undesirable artifact of the smoothing operation. However, the erosion operation can alleviate the problem. After performing the erosion operation for a few time steps, the height values of such grid cells are reduced to zero.

6. Merging water drops

When water drops and water flows come into contact, they merge into a new water drop or water flow. To determine whether or not they touch each other, collision detection is performed. One may adopt the sphere–sphere intersection test for collision detection due to its simplicity. However, the shapes of water drops/flows are generally irregular. Thus, the sphere–sphere intersection test is not suitable. Our method employs ID map I to merge water

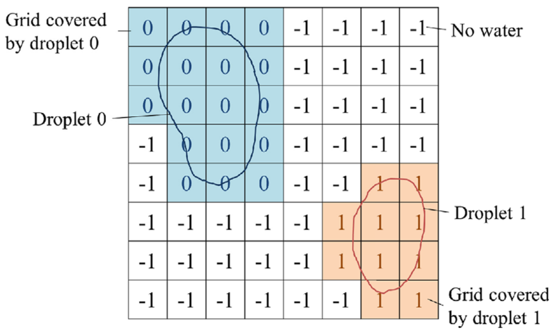

Fig. 12. ID map records the grid cells belonging to each water drop.

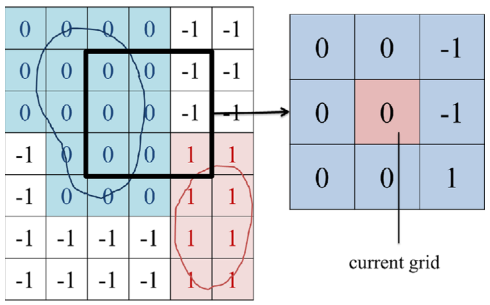

Fig. 13. The regions of two water drops are adjacent. These two water drops will merge.

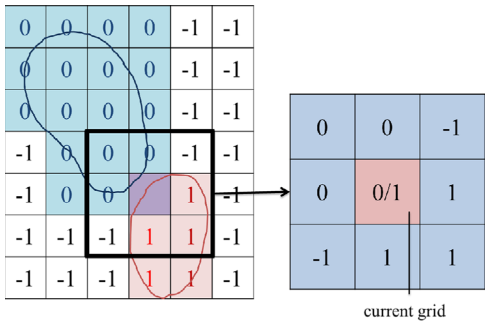

Fig. 14. The water drop with ID 0 and the water drop with ID 1 occupy one grid cell (0/1). The regions of the two water drops overlap with each other, and hence, the two water drops will merge.

drops and water flows. Fig. 12 shows an example of an ID map. A value of 0 at a grid cell indicates that the grid cell is covered by a water drop with ID 0.

The regions of two water drops i_0 and i_1 overlap with each other if they occupy at least one common grid cell. Their regions are adjacent if there exist (x_0, y_0) and (x_1, y_1) such that the following two conditions are satisfied:

1. $I(x_0, y_0) = i_0$ and $I(x_1, y_1) = i_1$;
2. $|x_0 - x_1| \le 1$ and $|y_0 - y_1| \le 1$.

Two water drops merge into a new water drop if their regions are adjacent or overlap with each other. A new particle is created to represent the new water drop. We illustrate two cases of merging in Figs. 13 and 14. Fig. 13 shows that the regions of two water drops are adjacent. Fig. 14 shows that the regions of two water drops overlap as the drops move.

After two water drops merge, the velocity of the new water drop is computed as $\mu(m_{i_0}\mathbf{v}_{i_0} + m_{i_1}\mathbf{v}_{i_1})/(m_{i_0} + m_{i_1})$, where $\mathbf{v}_{i_0}$ and $\mathbf{v}_{i_1}$ are the velocities of the two water drops; and μ is a user defined value. For the conservation of linear momentum, μ is 1. However, we set μ to be greater than 1 in our experiments because we observe that the new water drop moves more rapidly than the two merged water drops. Most of the water moves to the lower part of the new water drop. Hence, we set the position of the new particle to the lowest position of the two water drops. Similarly, we can handle the merging of multiple water drops. The mass of the new water drop is the sum of the masses of all of the merged water drops. Fig. 15 shows the simulation result obtained for the merging of two water drops.

7. Implementation and results

We implemented the physical simulation layer on a CPU and the rendering layer on a GPU. The simulation time step was 0.005 s. Our experiments were performed on an Intel(R) Core (TM) i7-2600 CPU at 3.40 GHz quad core with 16 GB RAM and NVIDIA GeForce GTX670. The code for the physical simulation was written in C++, and the code for rendering was written in the GLSL shading language. Fig. 16 shows our rendering process. The normal map is computed based on the height map. The normal vector $\mathbf{n}(x, y)$ of a point at the position (x, y) of the normal map is computed as

$$\begin{aligned}\mathbf{n}(x,y) = &(\ell, 0, H(x+1,y)) - (-\ell, 0, H(x-1,y)) \\ &\times (0, \ell, H(x,y+1)) - (0, -\ell, H(x,y-1)) \\ = &(2\ell, 0, H(x+1,y) - H(x-1,y)) \\ &\times (0, 2\ell, H(x,y+1) - H(x,y-1))\end{aligned}$$

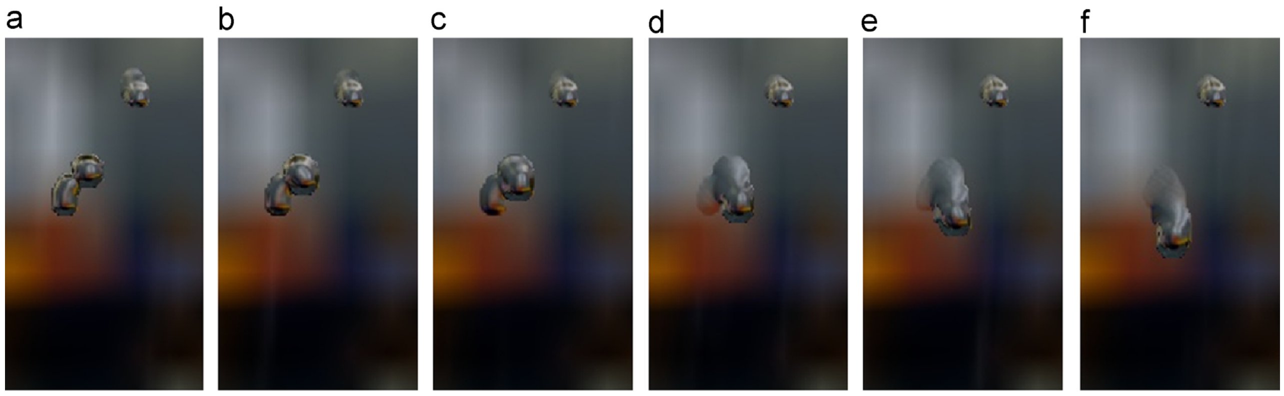

Fig. 15. Simulation result for the merging of two water drops.

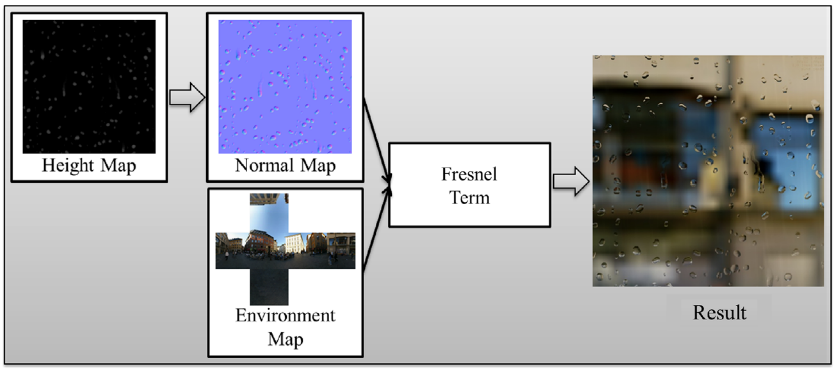

Fig. 16. Rendering pipeline.

Table 1
Timings of the different parts of the simulation.

Height map resolution	250 water drops		500 water drops	
	Part A (ms)	Part B (ms)	Part A (ms)	Part B (ms)
500^2	4–5	1–4	8–9	4–7
1000^2	7–8	2–4	11–12	4–8

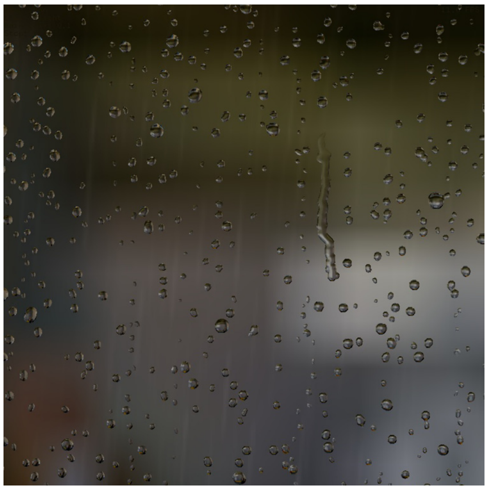

Fig. 17. Our simulation result for water droplets on a glass pane. The height map resolution is 512×512.

Then, we normalize the normal vector of each point in the normal map, i.e., $\mathbf{n}(x,y)/\|\mathbf{n}(x,y)\|$. Finally, the result is rendered with the environment map and Fresnel term.

The timing performance of our method can be divided into two parts:

1. Part A: computation of height map, updating height map and computation of ID map.
2. Part B: external force computation, computation of movement direction, computation of residual droplets, computation of the shape of droplets, and merging water drops and water flows.

Table 1 shows the timings of the different parts of the simulation. For a 500×500 grid and 500 water drops, our system required approximately 12–16 ms for the physical simulation and approximately 3 ms for rendering. One of the simulated results is shown in Fig. 17. We compare our simulated results with the real-world examples in the real world, as shown in Fig. 18. Our system can render water drop shapes that are similar to those observed in the real world. Fig. 19 shows other examples with different background images.

7.1. Discussion and limitations

We propose a simple method for simulating water drops and water flows. Although we do not fully consider their physical properties, our method can simulate water drops and water flows realistically. We use an ID map for merging water drops. When two water drops merge, one is absorbed by another. The height map records the amount of water in the absorbed drop in the previous time step. Then, we apply smoothing and erosion operations to update the height map. Hence, the process of merging water drops is continuous in our method. Our method does not extensively account for the physical properties of water. Thus, the method is not able to accurately compute the contact angle between the fluid and the solid surface. The advantage of our proposed method over the level-set method is that our method features simple computation and can achieve results at real-time rates.

Our method is restricted to 2D grids, and the predominant direction of movement of water drops and water flows should align with the direction along which gravity acts. Because our method is a particle-based approach, the mass of the entire system is conserved. However, our method does not conserve the volume of water due to the smoothing and erosion operations. We do not consider the amount of water in updating the height field. The rate of shrinking of water trails is dependent on the grid resolution. With the same simulation interval, a lower-resolution grid will cause water trails to shrink more rapidly. We may observe artifacts, such as some amount of water disappearing gradually, due to the smoothing and erosion operations. We can alleviate this problem by adopting a height map with adaptive resolution. Our method adopts hemispheres as the primitive shape for modeling the shape of water drops and flows. For fast running water drops, the shapes of hemispheres are still visible. In the future, we would like to develop a technique to stretch the primitive shape along the directions of movement of water drops.

8. Conclusion and future works

We have presented a novel approach for modeling the motion behavior of water drops and water flows. Our system combines both a particle system and a height map. We employ an ID map to merge water drops efficiently even though the shapes of water drops and water flows are irregular. We also propose a smoothing operation and an erosion operation for dynamically updating the

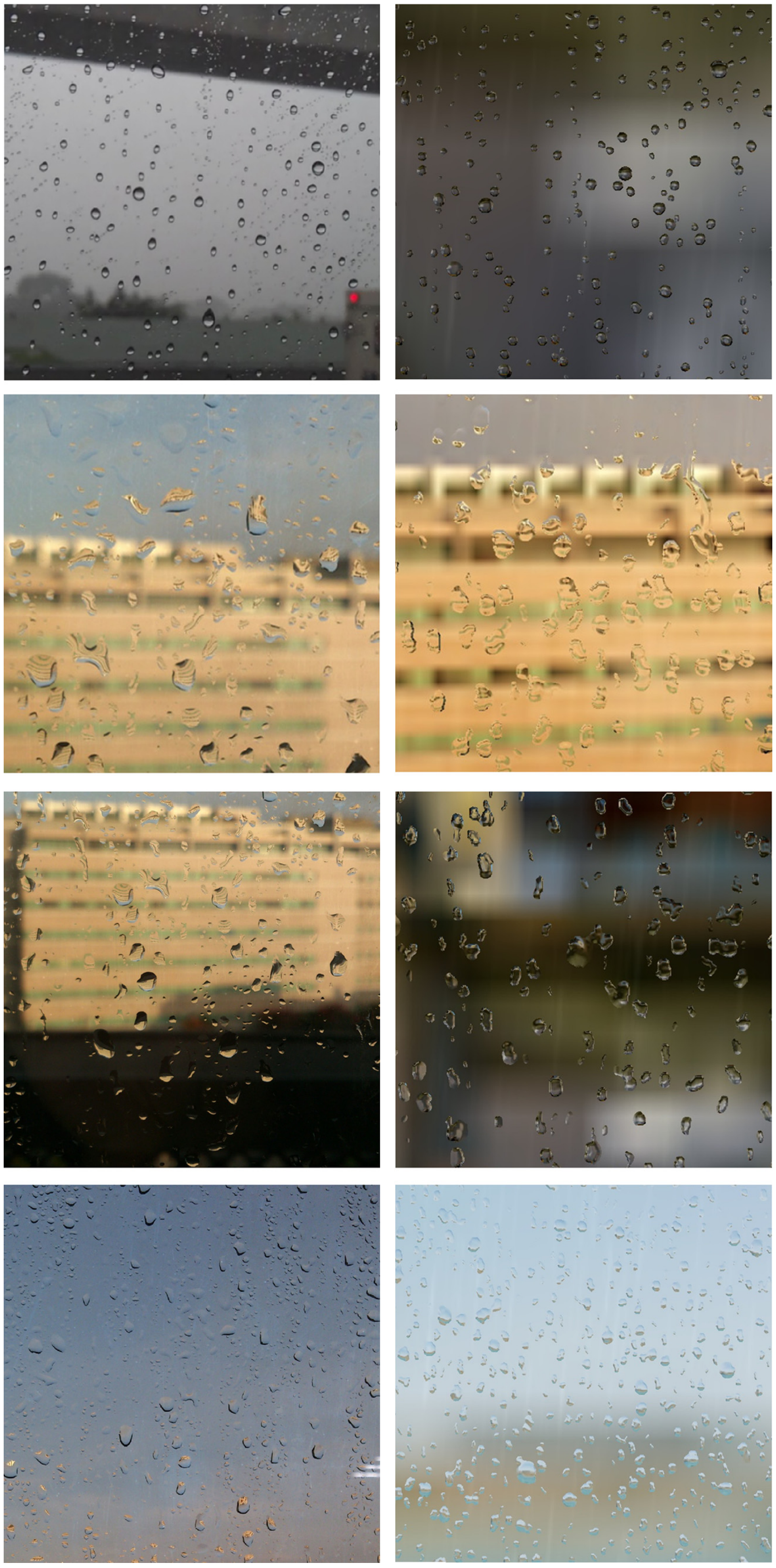
Fig. 18. Comparison between water drops captured in the real world (left) and our simulated results (right).

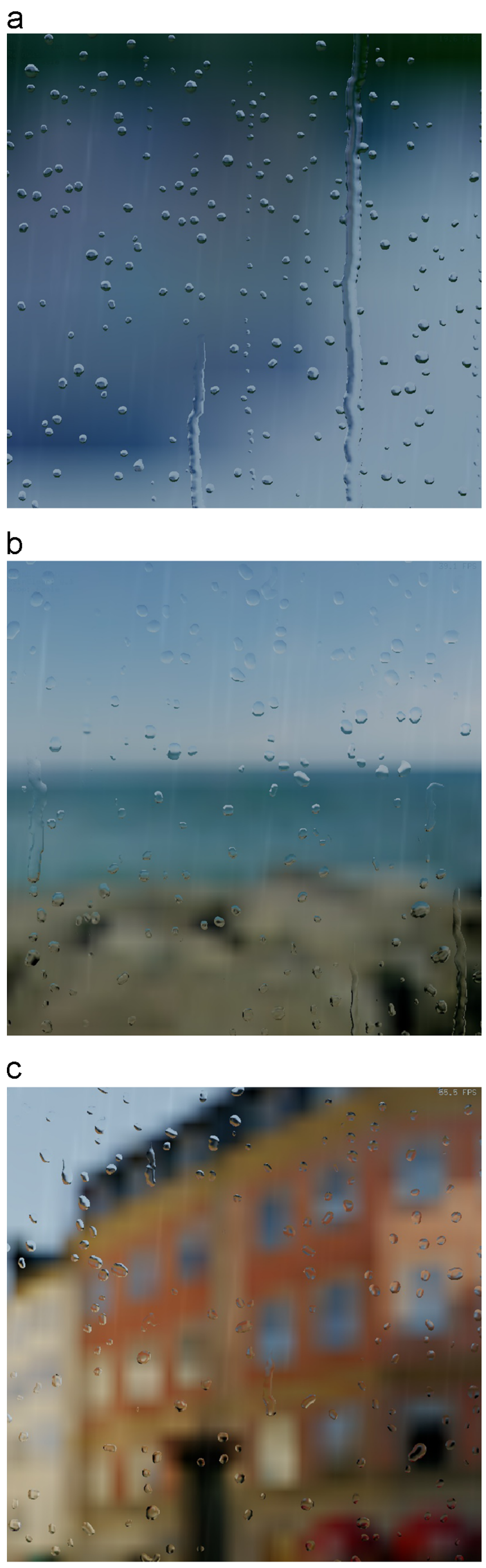

Fig. 19. Rendering results with different background images.

height map to smooth and shrink water flows inward. Our method is simple and easy to implement. It can be applied to applications such as driving simulations and video games that require fast performance. In the future, we would like to extend our method to simulate water drops on the surfaces of three-dimensional objects. Furthermore, we would like to implement our method on multi-core platforms so that a smaller time step can be used to produce more natural movement of water drops and water flows. We would also like to develop techniques to improve the quality of the contours of water drops and water flows.

Acknowledgments

We would like to thank the useful comments of the reviewers for improving the presentation of the paper. This work was supported in part by the National Science Council of ROC (Taiwan) NSC 101-2221-E-009-157 and NSC 102-2218-E-009-003.

Appendix A. Supplementary data

Supplementary data associated with this paper can be found in the online version of http://dx.doi.org/10.1016/j.cag.2013.08.004.

References

[1] Algan E, Kabak M, Ozguc B, Capin T. Simulation of water drops on a surface. In: 3DTV conference: the true vision—capture, transmission and display of 3D video, 2007. p. 361–4.
[2] Chen K-C, Chen P-S, Wong S-K. A hybrid method for water droplet simulation. In: ACM SIGGRAPH VRCAI, 2012. p. 341–4.
[3] Fournier P, Habibi A, Poulin P. Simulating the flow of liquid droplets. In: Proceedings of graphics interface, 1998. p. 133–42.
[4] Hirvi J, Pakkanen T. Nanodroplet impact and sliding on structured polymer surfaces. Surface Science 2008;602:1810–8.
[5] Jonsson M, Hast A. Animation of water droplet flow on structured surfaces. In: SIGRAD, 2002. p. 17–22.
[6] Kaneda K, Kagawa T, Yamashita H. Animation of water droplets on a glass plate. In: Proceedings of Computer Animation, 1993. p. 177–89.
[7] Kaneda K, Ikeda S, Yamashita H. Animation of water droplet moving down a surface. Journal of Visualization and Computer Animation 1999;10(1):15–26.
[8] Kaneda K, Zuyama Y, Yamashita H, Nishita T. Animation of water droplet flow on curved surfaces. In: Proceedings of Pacific graphics, 1996. p. 50–65.
[9] Nakata N, Kakimoto M, Nishita T. Animation of water droplets on a hydrophobic windshield. In: WSCG2012, 2012.
[10] Sato T, Dobashi Y, Yamamoto T. A method for real-time rendering of water droplets taking into account interactive depth of field effects. In: Entertainment Computing: Technologies and Applications, IFIP First International Workshop on Entertainment Computing (IWEC 2002), IFIP Conference Proceedings, vol. 240, 2002. p. 125–32.
[11] Stuppacher I, Supan P. Rendering of water drops in real-time. In: Central European seminar on computer graphics for students (CESCG 2007), 2007.
[12] Takenaka S, Mizukami Y, Tadamura K. A fast rendering method for water droplets on glass surfaces. In: The 23rd international technical conference on circuits/systems, computers and communications, 2008. p. 13–6.
[13] Tatarchuk N. Artist-directable real-time rain rendering in city environments. Marlborough, MA, USA: ATI Research Inc., 3D Application Research Group; 2006.
[14] Wang H, Miller G, Turk G. Solving general shallow wave equations on surfaces. In: Proceedings of ACM SIGGRAPH/Eurographics symposium on computer animation, 2007. p. 229–38.
[15] Wang H, Mucha PJ, Turk G. Water drops on surfaces. Transaction on Graphics 2005;24(3):921–9.
[16] Yu YJ, Jung H-Y, Cho H-G. A new water droplet model using metaball in the gravitational field. Computer and Graphics 1999;23:213–22.
[17] Yvonne J, Johannes B. Gpu-based real-time on-surface droplet flow in x3d. In: Proceedings of the 14th international conference on 3D web technology, 2009. p. 51–4.
[18] Zhang Y, Wang H, Wang S, Tong Y, Zhou K. A deformable surface model for real-time water drop animation. IEEE Transactions on Visualization and Computer Graphics 2012;18(August (8)):1281–9.