Animation of Water Droplet Flow on Curved Surfaces

Kazufumi Kaneda, Yasuhiko Zuyama, Hideo Yamashita
Faculty of Engineering, Hiroshima University
1-4-1 Kagamiyama, Higashi-hiroshima, 739 Japan
E-mail: kin@eml.hiroshima-u.ac.jp
and
Tomoyuki Nishita
Faculty of Engineering, Fukuyama University
985 Higashimura-cho, Fukuyama, 729-02 Japan
E-mail: nis@eml.hiroshima-u.ac.jp

Abstract

This paper proposes a method for generating a realistic animation of water droplets and their streams on curved surfaces, such as a windshield, taking into account the dynamics that act on the droplets. A water droplet runs down an inclined surface. The stream meanders down because of impurities on the surface. Some amount of water remains behind because of the nature of the wetting phenomena. Therefore, the mass of the droplet decreases, and the droplet is decelerated as the result. Finally, the flow stops.

In this paper, a discrete model of curved surfaces is developed to simulate the streams from the water droplets as described above. The curved surface is divided into small meshes, and the flow of water droplets is calculated based on probability of movement. For rendering scenes to a wide variety of applications, we also develop two rendering methods: a fast rendering method using a simple model of water droplets and a high quality rendering method that pursues the photo-reality. Animations of water droplets on a windshield or a teapot demonstrate the usefulness of the proposed method.

Keywords: Animation, Water Droplet Flow, Curved Surface, Discrete Model, Dynamics

1 Introduction

Since the 1980's, many methods have been developed for modeling and rendering water and other fluids. Methods of modeling ocean waves have been proposed [Max 81, Mastin 87, Peachey 86, Fournier 86, Ts'o 87]. To improve the reality, the color of water surfaces calculated taking into account the radiative transfer of light [Kaneda 91] and underwater glittering due to lens effect [Watt 90] has also been rendered. Despite these advances, many problems remain unaddressed in rendering water because of its multiformity and the complex motion.

The animation of water droplets on a curved surface is necessary for many applications (e.g., drive simulators). The shape and motion of water droplets on a surface are under the sway of such factors as gravity, interfacial tension, air resistance, and so on. Much effort in the field of physics has been devoted to understanding the interactions between liquids and solids [Gennes 85, Janosi 89].

Most of the water models developed for computer graphics attempt primarily to render large bodies of water without boundaries, such as the ocean [Max 81, Mastin 87]. They typically avoid realistic scenes containing a seashore, where a boundary between liquid and solid must be depicted. By taking into account the motion of water near boundaries, the realism of water animations has been improved [Peachey 86, Fournier 86, Ts'o 87]. However, these methods of modeling water cannot realistically animate the flow of very small amounts because particles of water move in circular or ellipsoidal orbits around their initial positions in the models. Recently, several methods for rendering the flow of small amounts of water have been developed [Kass 90, Kharitonsky 93, Chiba 95, Mallinder 95]. These include a method of modeling shallow water taking into account fluid dynamics [Kass 90], and a physically based model for simulating icicle growth [Kharitonsky 93]. Employing particle systems [Reeves 83], water currents were rendered realistically [Chiba 95]. To save memory for storing information on particles, a method called string textures was proposed, allowing the rendering of large waterfalls [Mallinder 95]. However, it is difficult to animate the streams of water drops on curved surfaces using these models. To address this problem, we previously developed a method for simulating the flow of water droplets on a flat glass plate [Kaneda 93]. However, the previous method cannot simulate a water droplet on curved surfaces, an indispensable technique in drive simulators.

This paper proposes an extended method for generating realistic animation of water droplets and their streams on curved surfaces, taking into account the dynamics that act on water droplets. The motion of water droplets on a surface depends on an external force, such as gravity and wind, and the interfacial tension between the water and the surface. Water droplets run down an inclined surface when the force that acts on the droplet exceeds a static critical force. They also tend to meander due to the roughness of the surface, microscopical impurities on the surface, etc. The mass of the droplet decreases as it runs, because some amount of water remains behind the flow due to wetting. Eventually, the force acting on the droplet becomes weaker, and the flow stops.

To simulate the flow of water droplets described above, this paper proposes a discrete surface model of curved surfaces. In the model, water droplets move from one grid to the next on the discrete surface. It is quite difficult to simulate the flow of water droplets for the purpose of high-precision engineering, because such a flow is a complicated process in which many parameters play a role [Gennes 85]. Our main purpose is to generate a realistic animation, taking into account the dominant parameters of dynamic systems: gravity to water droplets, interfacial tensions, and water merging.

For rendering scenes containing water droplets to a wide variety of applications, we also propose two rendering methods taking into account reflection and refraction of light: a fast rendering method using a simple model of water droplets and a high quality rendering that pursues the photo-reality. Animations of water droplets on a windshield or a teapot demonstrate the usefulness of the proposed method.

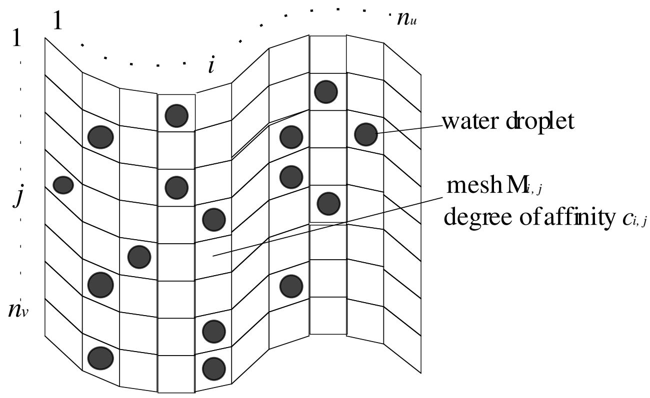

Figure 1: Discrete model of a curved surface.

2 Simulation of Water Droplet Flow

Water droplet flow on curved surfaces is a very complicated phenomenon. Unsolved flow mechanisms remain along with many unmeasured parameters with respect to interfacial dynamics between fluids and solids. The proposed method of simulating the flow of water droplets on curved surfaces is based on dominant elements within the dynamics because our main purpose is the generating of realistic animations that are useful for many kinds of simulators (e.g., drive simulators).

2.1 The Outline of the Proposed Method

A water droplet begins to run down an inclined surface when the force that acts on the droplet exceeds a critical strength. The route of the stream is roughly determined by the force of gravity and wind, but it meanders down the surface because of impurities on the surface. Some amount of water remains behind because of the wetting phenomenon. Therefore, the mass of the droplet decreases and the force acting on the droplet becomes weaker than the resistance to movement. The droplet decelerates, and the flow eventually stops.

In order to simulate the flow of water droplets as described above, the curved surface is divided into small quadrilateral meshes, and water droplets travel from one mesh point to the next. Figure 1 shows a discretized surface that has a $n_u \times n_v$ lattice on the curved surface. The curved surface may be specified by Bezier patches. We convert the Bezier patches into a discrete surface model. That is, each quadrilateral mesh has coordinates of the four vertices and a coordinate of the center. It also has a normal vector at the center of the mesh. Although all of the four vertices of a quadrilateral mesh do not lie on the same plane, in general, the quadrilateral mesh is approximated as a plane because of the minute size, and the normal of the approximated plane is used as that of the mesh. The movement of a water droplet is calculated on the approximated plane of the mesh in which the water droplet is located.

Degree of affinity for water, $c_{i,j}$ $(0 \le c_{i,j} \le 1)$, is assigned for each mesh $\mathrm{M}_{i,j}$ in advance. This describes the lack of uniformity on an object surface because of impurities and small scratches. In most cases, the degree of affinity is assigned randomly based on a normal distribution to contribute meanders of the streams and the wetting phenomena.

The process for simulating the flow of water droplets on a curved surface is as follows:

(1) Make a discretized surface model, and specify degree of affinity for each mesh as a pre-process of a flow simulation.

(2) Put new water droplets at mesh points, and specify a weight and initial speed for each droplet. Water droplets may be put at any mesh points, depending on applications. In the case of animation for drive simulators, water droplets are dispersively fed in every frame of the animation.

(3) Initialize timekeepers for all water droplets. The timekeeper accumulates times taken for a droplet to travel from one mesh to another. When the accumulated time reaches a frame time (in the case of a NTSC video animation, 1/30 seconds), the water droplet is frozen.

(4) Check whether a water droplet moves or not. If it moves, follow the next step. If not, changing a droplet, repeat this step until all of the droplets are processed.

(5) Determine the mesh the droplet will visit next.

(6) Calculate the time required for the droplet to move to the next mesh.

(7) Move the droplet to the next mesh, if the timekeeper allows the droplet to move.

(8) Repeat steps 4 through 7, until all of the droplets are frozen.

(9) Repeat steps 2 through 8 for the duration of animation.

In the following sections, we discuss the details of steps 4 through 7. These steps are essential to the proposed method.

2.2 Movement Test

Consider a water droplet at a mesh point $\mathrm{M}_{i,j}$ and a normal vector, $\boldsymbol{N}_{i,j}$, at the mesh. The droplet receives external forces: the force of gravity and the force of wind in our method. Let's assume projected force vectors of gravity and wind onto the plane whose normal is $\boldsymbol{N}_{i,j}$ are $\boldsymbol{f}_{g(i,j)}$ and $\boldsymbol{f}_{w(i,j)}$, respectively. When the external force, $\boldsymbol{f}^{\mathrm{ext}}_{i,j}$, composed of $\boldsymbol{f}_{g(i,j)}$ and $\boldsymbol{f}_{w(i,j)}$ exceeds a critical force, the water droplet start to move:

$$\left| \boldsymbol{f}^{\mathrm{ext}}_{i,j} \right| > F_{i,j}, \tag{1}$$

where $\boldsymbol{f}^{\mathrm{ext}}_{i,j} = \boldsymbol{f}_{g(i,j)} + \boldsymbol{f}_{w(i,j)}$, and $F_{i,j}$ is a static critical force. The static critical force is the resistance that prevents a droplet from moving, and originates from the interfacial tension between water and a surface. The degree of affinity set to each mesh depends on the

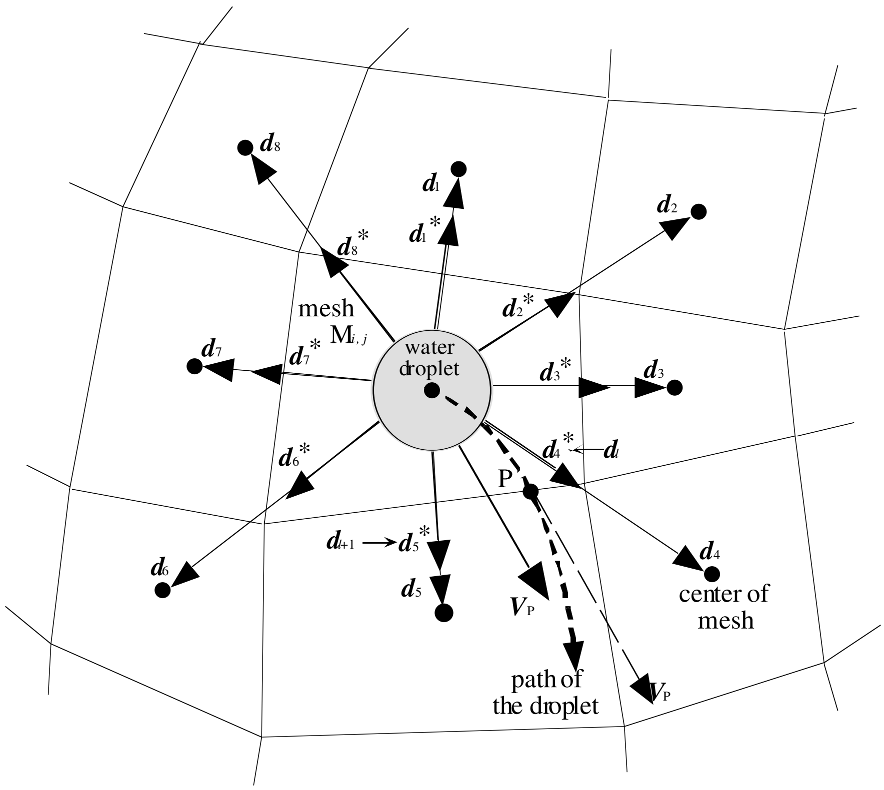

Figure 2: Determining direction of movement.
All of the coordinates and vectors corresponding to the neighboring meshes are projected onto the approximated plane of mesh $M_{i,j}$.
$\boldsymbol{d}_k^*$ (k = 1, ..., 8) is an unit vector, and $\boldsymbol{V}_P$ is a principal vector.

interfacial tension because it expresses the condition of a surface, such as impurities and small scratches. Therefore, in the proposed method, the static critical force is calculated by using the degree of affinity, $c_{i,\,j}$.

$$F_{i,\,j} = \beta_s\, c_{i,\,j}, \qquad (2)$$

where β_s is a coefficient for converting the degree of affinity into the resistance in a static state, and is set experimentally.

2.3 Direction of Movement

The route of the stream is roughly determined by the external forces acting on the water droplet, but it meanders down the surface because of impurities and small scratches on the surface. In the proposed method, direction of movement is determined by using a roulette to achieve a natural stream as described above. That is, we classify direction of movement into the eight directions, $\boldsymbol{d}_k^*$ ($k = 1, 2, \cdots, 8$), that point to the center of the neighboring eight meshes (see Fig. 2), and calculate probabilities for each direction. The probabilities literally express the probability of a water droplet visiting the mesh corresponding to a given direction, and the areas of each domain in the roulette are proportional to the probabilities. Using the roulette and a random number, direction of movement is selected from the eight directions.

Although the number of the discrete directions is limited, the global stream can flow in any direction, because the movement of water droplets is determined by using probability. This is similar to the way a digital xy-plotter, with only limited directions of movement, can draw various lines and curves. At the same time, the method makes it possible to simulate the randomness of streams of water droplets.

2.3.1 Making a Roulette

The probabilities, i.e., the areas of each domain in a roulette, are calculated based on the following three factors:

(1) Direction of movement under circumstances in which it obeys Newton's law of motion

(2) Degree of affinity for water on the neighboring meshes

(3) Condition (wet or dry) of the neighboring meshes

<u>Probability based on Newton's law of motion</u>

If it is assumed that the movement of a water droplet on a curved surface obeys Newton's law of motion, and there is no resistance to the movement, then the path of the droplet in mesh $M_{i,j}$ can be expressed by Newton's equation of motion:

$$\frac{d^2\boldsymbol{x}(t)}{dt^2} = \frac{\boldsymbol{f}^{\text{ext}}_{i,j}}{m_{i,j}}, \tag{3}$$

where $\boldsymbol{x}(t)$ is the position of the droplet at time t, $m_{i,j}$ is the mass of the water droplet, and $\boldsymbol{f}^{\text{ext}}_{i,j}$ is the external force. In our method, the external force is calculated by using the forces of gravity and wind; $\boldsymbol{f}^{\text{ext}}_{i,j} = \boldsymbol{f}_{g(i,j)} + \boldsymbol{f}_{w(i,j)}$. Solving Eq. 3, we can find position P, where the droplet crosses the boundary of the mesh $M_{i,j}$, and also calculate droplet velocity $\boldsymbol{V}_{\mathrm{P}}$, when it arrives at the crossing position (see Fig. 2).

Probability based on Newton's law of motion is calculated using the velocity vector, $\boldsymbol{V}_{\mathrm{P}}$, called a principal vector in the discussion below. The probabilities are distributed between the two directions nearest to the principal vector, $\boldsymbol{V}_{\mathrm{P}}$, when the principal vector is located at the center of the mesh, $M_{i,j}$. If the nearest two directions are $\boldsymbol{d}^*_{\ell}$ and $\boldsymbol{d}^*_{\ell+1}$, the probability based on Newton's law of motion is determined by the following equation.

$$D_k = \begin{cases} \dfrac{1}{D_{\mathrm{sum}}} \left(\boldsymbol{d}_k^* \bullet \boldsymbol{V}_{\mathrm{P}} \right) & (k = \ell, \ell + 1) \\ 0 & (\text{otherwise}) \end{cases}, \tag{4}$$

$$D_{\mathrm{sum}} = \left(\boldsymbol{d}_\ell^* \bullet \boldsymbol{V}_{\mathrm{P}} \right) + \left(\boldsymbol{d}_{\ell+1}^* \bullet \boldsymbol{V}_{\mathrm{P}} \right),$$

where D_k ($k = 1, 2, \cdots, 8$) is a probability based on Newton's law of motion, $\left(\boldsymbol{d}_k^* \bullet \boldsymbol{V}_{\mathrm{P}} \right)$ expresses an inner product between two vectors $\boldsymbol{d}_k^*$ and $\boldsymbol{V}_{\mathrm{P}}$, and $\boldsymbol{d}_k^*$ is a unit direction vector toward the center of the neighboring mesh (see Fig. 2).

Probability based on affinity

Streams of water droplets meander because of impurities and small scratches on the surface. To achieve this effect, we take into account the degree of affinity assigned to the neighboring meshes when determining the mesh the water droplet will visit next.

Let's assume that the degree of affinity in the neighboring mesh corresponding to direction $\boldsymbol{d}_k^*$ is c_k ($k = 1, 2, \cdots, 8$), and a principal vector is $\boldsymbol{V}_{\mathrm{P}}$. Probabilities based on affinity are distributed to the directions whose angle from the principal vector, $\boldsymbol{V}_{\mathrm{P}}$, are smaller than 90 degrees, and are calculated by the following equation.

$$A_k = \frac{c_k}{A_{\mathrm{sum}}} \mathrm{u}\left(\left(\boldsymbol{d}_k^* \bullet \boldsymbol{V}_{\mathrm{P}} \right) \right), \tag{5}$$

$$A_{\mathrm{sum}} = \sum_{l=1}^{8} c_l \, \mathrm{u}\left(\left(\boldsymbol{d}_l^* \bullet \boldsymbol{V}_{\mathrm{P}} \right) \right),$$

where A_k ($k = 1, 2, \cdots, 8$) is a probability based on affinity, and $\mathrm{u}(x)$ is the unit step function, i.e. if $x \geq 0$, then $\mathrm{u}(x) = 1$; otherwise, $\mathrm{u}(x) = 0$.

Probability based on wet or dry conditions

Water droplets tend to follow a stream already present. This phenomenon shows that a water droplet moves to wet places with high probability. To take this effect into account, the proposed method examines whether the neighboring meshes are wet or dry. If a water droplet exists on a neighboring mesh, a higher probability is set for that direction. To prevent a water droplet from suddenly changing the direction of movement to the opposite direction, the probabilities are shared only with the directions whose angle from the principal vector, $\boldsymbol{V}_{\mathrm{P}}$, are smaller than 90 degrees.

$$W_k = \frac{g\left(\boldsymbol{d}_k^* \right)}{W_{\mathrm{sum}}} \mathrm{u}\left(\left(\boldsymbol{d}_k^* \bullet \boldsymbol{V}_{\mathrm{P}} \right) \right), \tag{6}$$

$$W_{\text{sum}} = \sum_{l=1}^{8} g\left(\boldsymbol{d}_l^*\right) \mathrm{u}\left(\left(\boldsymbol{d}_l^* \bullet \boldsymbol{V}_{\mathrm{P}}\right)\right),$$

where W_k ($k = 1, 2, \cdots, 8$) is a probability based on wet or dry conditions, and $g\left(\boldsymbol{d}_k^*\right)$ is a function to check the wet or dry conditions. That is, the function gives 1 when there is a water droplet on the neighboring mesh corresponding to direction $\boldsymbol{d}_k^*$, and gives 0 when there is no water droplet. If $W_{\text{sum}} = 0$, W_k ($k = 1, 2, \cdots, 8$) is set to zero.

Roulette areas

The probability used for determining the direction of movement, i.e., the area of a roulette, is determined by using the three factors described above. The greater the speed of the droplet, the higher the probability of moving toward the principal vector, $\boldsymbol{V}_{\mathrm{P}}$. Taking this effect into account, the probability based on Newton's law of motion is weighted by the length of the principal vector.

$$R_k = \frac{\alpha_1 \left| \boldsymbol{V}_{\mathrm{P}} \right| D_k + \alpha_2 A_k + W_k}{R_{\text{sum}}}, \tag{7}$$

$$R_{\text{sum}} = \sum_{l=1}^{8} \left(\alpha_1 \left| \boldsymbol{V}_{\mathrm{P}} \right| D_l + \alpha_2 A_l + W_l \right),$$

where R_k ($k = 1, 2, \cdots, 8$) is the area of each domain in the roulette, and α_1 and α_2 are parameters for controlling the regularity and irregularity (meanders), respectively.

2.3.2 Determining Direction of Movement

Using the roulette described above, the direction of movement, i.e. the mesh a water droplet will visit next, is determined. A random number, r ($0 < r \leq 1$), is generated based on the uniform distribution, and the index, p ($p = 1, 2, \cdots, 8$), that satisfies the following conditions is sought.

$$\sum_{k=1}^{p-1} R_k < r \leq \sum_{k=1}^{p} R_k \,, \tag{8}$$

where $\sum_{k=1}^{0} R_k = 0$. In this way, the direction of movement is determined as $\boldsymbol{d}_p^*$.

2.4 Calculating the Time required for Droplets to Travel

After determining the direction of movement, the water droplet is moved to the next mesh. In this step, the time taken for the droplet to travel to the next mesh has to be calculated, and the timekeeper accumulates the time. If the accumulated time exceeds a frame time, the droplet is not moved.

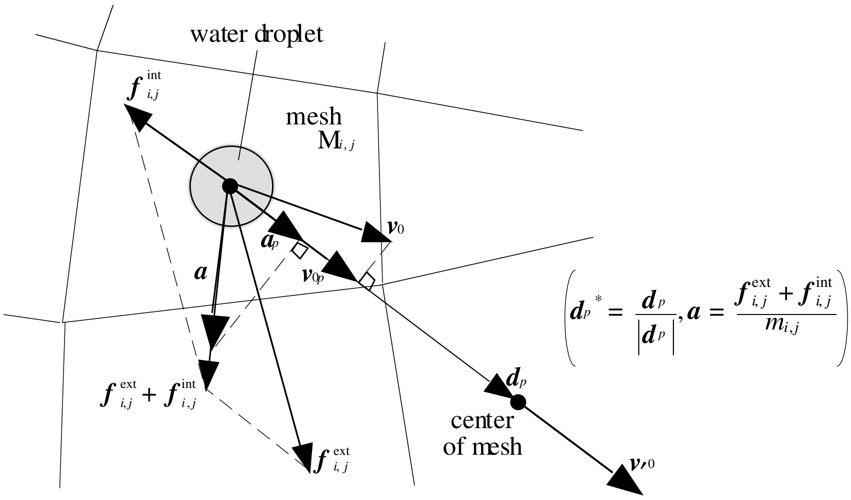

Figure 3: Projected acceleration and projected initial speed of a water droplet.

Let's assume $\boldsymbol{d}^*_p$ is the unit vector that indicates the direction of movement, and $\boldsymbol{d}_p$ is the movement vector that connects the center of the mesh the droplet will visit next (see Fig. 3). The time, t, taken for the droplet to travel to the next mesh is calculated by the following equation derived from Newton's equation of motion.

$$\frac{1}{2}\boldsymbol{a}_p t^2 + \boldsymbol{v}_{0\,p}\, t = \boldsymbol{d}_p, \tag{9}$$

where $\boldsymbol{a}_p$ and $\boldsymbol{v}_{0\,p}$ are an projected acceleration and projected initial speed of the water droplet, respectively. The acceleration is calculated by using the mass, $m_{i,\,j}$, of the water droplet and the force that acts on the droplet. The acceleration is projected onto the movement vector, $\boldsymbol{d}_p$, to decompose the acceleration into the component toward direction of the movement. That is, the projected acceleration is calculated by the following equation.

$$\boldsymbol{a}_p = \frac{\left(\left(\boldsymbol{f}^{\mathrm{ext}}_{i,\,j} + \boldsymbol{f}^{\mathrm{int}}_{i,\,j}\right) \bullet \boldsymbol{d}^*_p\right)}{m_{i,\,j}}\, \boldsymbol{d}^*_p, \tag{10}$$

where $\boldsymbol{f}^{\mathrm{ext}}_{i,\,j}$ is an external force composed of the forces of gravity and wind, and $\boldsymbol{f}^{\mathrm{int}}_{i,\,j}$ is a force of resistance whose direction is opposite to the direction of movement, $\boldsymbol{d}^*_p$. The resistance originates from the interfacial tension between a water droplet and a surface. Therefore, it depends on the affinity, $c_{i,\,j}$, set to each mesh in advance. Taking into account these properties, the force of resistance is calculated using the following equation.

$$\boldsymbol{f}^{\mathrm{int}}_{i,\,j} = -\,\beta_{\mathrm{d}}\, c_{i,\,j}\, \boldsymbol{d}^*_p, \tag{11}$$

where β_d is a coefficient for converting the degree of affinity into the resistance in a dynamic state, and is set experimentally, satisfying the relationship $\beta_d < \beta_s$. If the external force becomes small, the inner product in Eq. 10 gives a negative strength. Consequently, the droplet is decelerated.

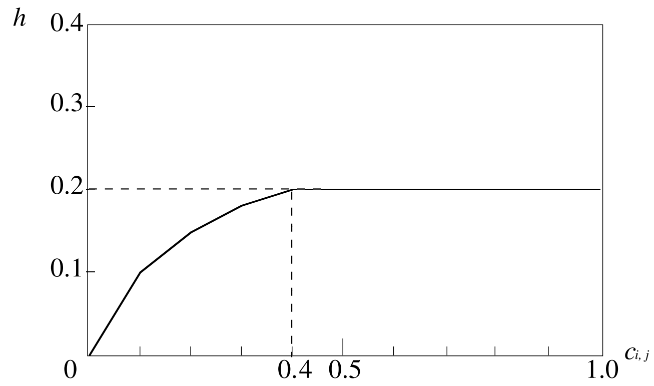

Figure 4: Function $h(c_{i,j})$ for determining a remaining water.

The water droplet has a initial speed, $\boldsymbol{v}_0$, when it is just traveling to the next mesh. The initial speed, $\boldsymbol{v}_0$, is also decomposed into the component toward direction of the movement, i.e. the projected initial speed, $\boldsymbol{v}_{0p}$.

$$\boldsymbol{v}_{0p} = \left(\boldsymbol{v}_0 \bullet \boldsymbol{d}_p^*\right) \boldsymbol{d}_p^*. \tag{12}$$

The initial speed is updated whenever the droplet arrives at the next mesh. Assuming the time for the droplet to travel to the next mesh is Δt and the movement energy corresponding to the initial speed perpendicular to movement, $\boldsymbol{v}_0 - \boldsymbol{v}_{0p}$, is consumed by the change of the movement direction, a new initial speed, $\boldsymbol{v}'_0$, for the droplet is calculated by the following equation.

$$\boldsymbol{v}'_0 = \boldsymbol{a}_p \Delta t + \boldsymbol{v}_{0p}. \tag{13}$$

2.5 Moving a Water Droplet

After deciding the direction of movement and calculating the time for the droplet to travel to the next mesh, the droplet is finally moved. When water flows on a surface, some amount of water remains behind because of the wetting, and the water poured in later merges into the water that remains behind. Therefore, in this step we have to take into account wetting and merging phenomena.

2.5.1 Wetting

When moving a water droplet, some amount of water remains behind. The mass of the remaining water, $m'_{i,j}$, i.e., the mass of a water droplet on mesh $M_{i,j}$ at the next step, depends on the affinity for water, $c_{i,j}$, and is found by the following equation.

$$m'_{i,j} = h\left(c_{i,j}\right) m_{i,j}, \tag{14}$$

where function $h\left(c_{i,j}\right)$ gives a coefficient for remaining water, and Fig. 4 shows the function, $h\left(c_{i,j}\right)$. The larger the affinity for water is, the more the remaining water. However, the function is saturated at the maximum coefficient.

2.5.2 Merging

If there is water on a mesh a water droplet enters, the water droplets merge into each other. In the proposed method, the speed of the new water droplet is determined by the law of conservation of momentum. Let's assume a water droplet on mesh $M_{i,j}$ moves to the neighboring mesh, $M_{i+k,j+l}$ (k, $l = -1, 0, 1$; not allow $k = l = 0$ at the same time), and the mass and the speed of the water droplet on mesh $M_{i+k,j+l}$ are $m_{i+k,j+l}$ and $\boldsymbol{v}_{i+k,j+l}$, respectively. The mass, $m'_{i+k,j+l}$, and the speed, $\boldsymbol{v}'_{i+k,j+l}$, of the new water droplet on the mesh $M_{i+k,j+l}$ are calculated by the following equations.

$$m'_{i+k,j+l} = m_{i+k,j+l} + m_{i,j} - m'_{i,j}, \tag{15}$$

$$\boldsymbol{v}'_{i+k,j+l} = \frac{m_{i+k,j+l}\,\boldsymbol{v}_{i+k,j+l} + \left(m_{i,j} - m'_{i,j}\right)\boldsymbol{v}'_0}{m'_{i+k,j+l}}, \tag{16}$$

where $\boldsymbol{v}'_0$ is obtained from Eq. 13.

Using the algorithm described above, the positions and masses of all water droplets on curved surfaces are calculated for every frame of animation.

3 Rendering Water Droplets on Curved Surfaces

Applications of water droplet animation are classified into two categories: those that principally pursue the rendering speed and those that pursue photo-reality. For example, drive simulators are classified as applications pursuing rendering speed because interactivity is vital for simulators. Of course, reality is important for any kind of application, but if the reality reaches a certain level, rendering speed becomes more important. On the other hand, many applications pursue photo-reality in the fields of entertainment, art, etc.

This paper proposes two methods for rendering water droplets on curved surfaces to serve both purposes. The first method is suitable for applications that pursue rendering speed; the other pursues the photo-reality.

3.1 A Fast Rendering Method Using a Simple Model

We developed a method for rendering water droplets on a flat glass plate [Kaneda 93]. The method employs an extended environment mapping to quickly render realistic images of water droplets, and uses spheres to make a simple model of water droplets by taking into account the contact angle of water and a solid surface. We have further extended the

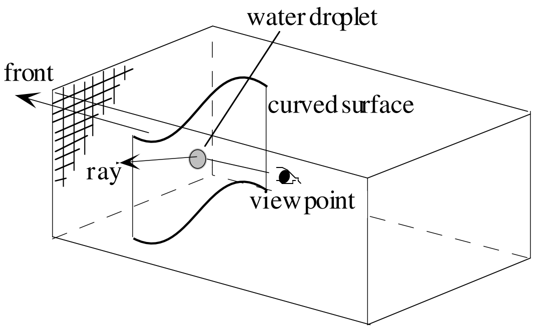

Figure 5: A cuboid for an extended environment mapping.

method to be able to handle Bezier surfaces. The outline of the extended method is as follows:

Step 1: Generating background textures for an extended environment mapping

The background textures are generated by projecting objects in the scene onto the faces of the cuboid that encloses the curved surfaces (see Fig. 5). Six textures are generated for the extended environment mapping.

Step 2: Tracing rays reflected or refracted by water droplets

In general, an incident ray splits into two directions, reflection and refraction, at the surface of a transparent object. However, in the case of water, one of these is the principal direction that greatly contributes the intensity of a pixel. Our renderings trace the rays only in the principal direction to save calculation time.

Thousands of water droplets are generated by the flow simulation described in the previous section. It is necessary to quickly find the first water droplet that a ray intersects. In the proposed method, the intersection between a ray and a curved surface is calculated first by using a Bezier clipping [Nishita 90], which gives an accurate intersection very quickly, because the water droplets are on curved surfaces. A water droplet located at the intersection is obtained quickly, if it exists, because of the discrete surface model in the flow simulation.

Step 3: Determining pixel colors

After tracing a ray, the face of the cuboid the ray intersects with is determined, and the intersection between the ray and one of the six faces is calculated. The pixel color corresponding to the ray is determined by using the background texture at the intersection.

3.2 A High Quality Rendering Method

A stream is just modeled as a group of spheres in the previous rendering method. Applications that pursue photo-reality must employ a more sophisticated model of water droplets. A method using meta-balls, so called blobby objects, is used for modeling the water droplets because they merge smoothly together in the model. The meta-ball is defined by its center, density distribution, and a threshold value for determining its surface. The center of a meta-ball is obtained from the flow simulation. The density distribution developed by Wyvill [Wyvill 90] is employed because of its favorable characteristics. That is, when two meta-balls that have the same volume coincide with each other, the volume becomes exactly twice that of the original.

The surface of the meta-ball is defined by the equi-density surface, the so called isosurface, whose density is the threshold value. Therefore, the surfaces of meta-balls are expressed by implicit equations, while curved surfaces, Bezier surfaces in our case, are expressed by parametric equations. It is difficult to render both surfaces with different expressions simultaneously. Conventional methods polygonalize meta-balls to avoid that difficulty, but doing so spoils the reality of the effect. To overcome this problem, we have developed a method that combines the two rendering methods: for Bezier surfaces [Nishita 90] and meta-balls [Nishita 94]. In the proposed method, the density distribution along a ray is converted into a Bezier function, and the intersection between the ray and the meta-ball is calculated using a Bezier clipping [Nishita 90], that is also employed when calculating the intersection between the ray and the curved surfaces.

When rays shooting from a view point arrive at the surfaces of transparent objects, they are traced in both directions as in conventional ray-tracing methods. The fast rendering method described above traces only the principal direction.

4. Examples

This section describes two examples, one for applications that pursue rendering speed, the other for those that pursue photo-reality. The examples are animations of water droplets on a windshield and the surface of a teapot, respectively.

4.1 Water Droplets on a Windshield

Figure 7 shows several frames of an animation rendered by the fast rendering method described in Section 3.1. A street scene is rendered from the driver's seat of a car. The windshield is modeled using a Bezier patch (see Fig. 6), and the average inclination of the windshield is 27 degrees. Several rain droplets are scattered in every frame of the animation. Images containing a wet road surface [Nakamae 90] are used as background textures. The resolution of the discrete surface model is 650×300, and the last image (the lower right image) contains about 32,500 water droplets. The calculation time depends on the number of the water droplets. The average time for the water flow simulation and the rendering per frame were about 1.5 seconds and 30 minutes, respectively, on SGI Power Indigo2 with a single R8000 CPU.

4.2 Water Droplets on the Surface of a Tea Pot

Figure 8 shows several frames of an animation rendered using the high quality rendering method described in Section 3.2. The tea pot is modeled using thirty two Bezier patches. Random amounts of water are fed at random positions in advance. The droplets whose

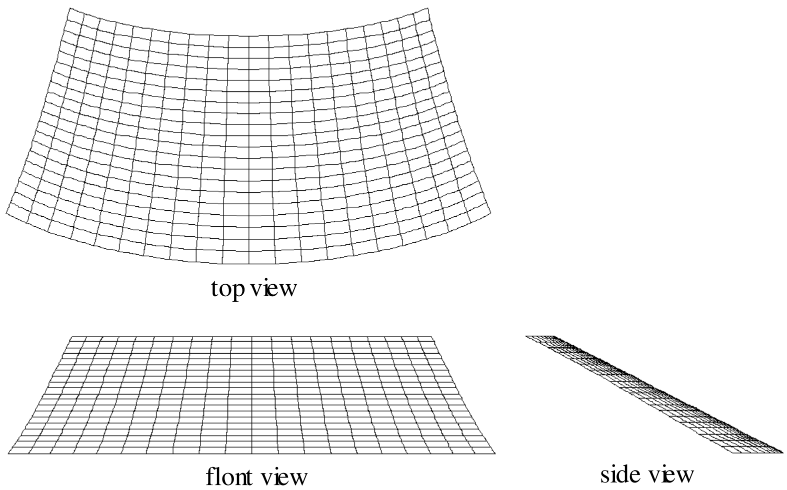

Figure 6: A windshield modeled using a Bezier patch.

external forces exceed the static critical force start to run down, and a running droplet merges with another which is lying in its path. The resolution of the discrete surface model is 400×400, and the last image (the lower right image) contains about 2,500 water droplets.

5. Conclusions

This paper proposes a method for generating realistic animation of water droplets and their streams on curved surfaces taking into account the dynamics that acts on the droplets. A discrete model of curved surfaces makes it possible to simulate the flow of water droplets running on the curved surface, and the use of a roulette to determine the direction of movement makes it possible to animate water droplets meandering on the curved surfaces.

Two rendering methods based on application purpose have been developed. A fast rendering method using a simple water droplet model can generate realistic animation at a reasonable cost. A high-quality rendering method makes it possible to render both water droplets and curved surfaces simultaneously, even though they are modeled in different expressions, i.e., meta-balls and Bezier patches. Animations of water droplets on a windshield or a teapot demonstrate the usefulness of the proposed method.

Several parameters are required to simulate the flow of water droplets, and these parameters play an important role in creating realistic animation. To pursue a more realistic animation, research into these parameters should be undertaken. The proposed method can simulate the flow of water droplets taking into account the effect of wind, but approximate forces were used in the examples because it is very difficult to calculate the exact force of wind acting on a droplet. It is also necessary to develop an effective method for calculating the force of the wind. Making use of graphics hardware and/or parallel computing, the proposed method becomes a promising technique in drive simulators.

References

[Chiba 95] Chiba N, Sanakanishi S, Yokoyama K, Ootawara I, Muraoka K, and Saito N. Visual Simulation of Water Currents Using a Particle-based Behavioural Model. *The Journal of Visualization and Computer Animation* **6**(3): 155-171 (1995).

[Gennes 85] de Gennes PG. Wetting: Statics and Dynamics. *Rev. Mod. Phys.* **57**(3): 827-863 (1985).

[Fournier 86] Fournier A. A Simple Model of Ocean Waves. *Computer Graphics* **20**(4): 75-84 (1986).

[Janosi 89] Janosi IM and Horvath VK. Dynamics of Water Droplets on a Window Pane. *Physical Review* **40**(9): 5232-5237 (1989).

[Kaneda 91] Kaneda K, Yuan G, Tomoda Y, Baba M, Nakamae E, and Nishita T. Realistic Visual Simulation of Water Surfaces Taking into Account Radiative Transfer. *Proc. Second International Conference Computer Aided Design & Computer Graphics*: 25-30 (1991).

[Kaneda 93] Kaneda K, Kagawa T, and Yamashita H. Animation of Water Droplets on a Glass Plate. *Proc. Computer Animation '93* : 177-189 (1993).

[Kass 90] Kass M and Miller G. Rapid, Stable Fluid Dynamics for Computer Graphics. *Computer Graphics* **24**(4): 49-57 (1990).

[Kharitonsky 93] Kharitonsky D and Gonczarowski J. A Physical Based Model for Icicle Growth. *The Visual Computer* **10**(2): 88-100 (1993).

[Mallinder 95] Mallinder H. The Modeling of Large Waterfalls using String Texture. *The Journal of Visualization and Computer Animation* **6**(1): 3-10 (1995).

[Mastin 87] Mastin GA, Watterberg PA, and Mareda JF. Fourier Synthesis of Ocean Scenes. *IEEE Computer Graphics & Applications* **7**(3): 16-23 (1987).

[Max 81] Max NL. Vectorized Procedural Models for Natural Terrain: Waves and Islands in the Sunset. *Computer Graphics* **15**(3): 317-324 (1981).

[Nakamae 90] Nakamae E, Kaneda K, Okamoto T, and Nishita T. A Lighting Model Aiming at Drive Simulators. *Computer Graphics* **24**(4): 395-404 (1990).

[Nishita 90] Nishita T, Sederberg TW, and Kakimoto M. Ray Tracing Rational Trimmed Surface Patches. *Computer Graphics* 24(4): 337-345 (1990).

[Nishita 94] Nishita T and Nakamae E. A Method for Displaying Metaballs by using Bezier Clipping. *Computer Graphics Forum* 13(3): 271-280 (1994).

[Peachey 86] Peachey DR. Modeling Waves and Surf. *Computer Graphics* **20**(4): 65-74 (1986).

[Reevs 83] Reevs WT. Particle Systems — A Technique for Modeling a Class of Fuzzy Objects. *ACM Transactions on Graphics* **2**(2): 91-108 (1983).

[Ts'o 87] Ts'o PY and Barsky BA. Modeling and Rendering Waves: Wave-Tracing Using Beta-Splines and Reflective and Refractive Texture Mapping. *ACM Transactions on Graphics* **6**(3): 191-214 (1987).

[Watt 90] Watt M. Light-Water Interaction using Backward Beam Tracing. *Computer Graphics* **24**(4): 377-385 (1990).

[Wyvill 90] Wyvill G and Trotman A. Ray-Tracing Soft Objects. *Proc. CG International '90* : 469-476 (1990).