The Visual Computer manuscript No.
(will be inserted by the editor)

Procedural and interactive icicle modeling

Jonathan Gagnon · Eric Paquette

Received: date / Accepted: date

Abstract Icicles formation is a complex phenomenon which makes it difficult to model for computer graphics applications. The methods commonly used in computer graphics to model icicles provide only minimal control over the results and require several minutes or even hours of computation. This paper proposes a procedural approach allowing interactive modeling, which is broken down into four stages. The first computes the water motion on the surface; the second determines where the water drips; the third computes the trajectories of the icicles growth, and the fourth creates the surface. In addition, the approach allows the creation of glaze ice. The results are not only realistic but also rapidly computed. This approach provides a significant increase in control over results and computation speed.

Keywords Natural phenomena · Procedural modeling · Icicle · Ice modeling · Glaze ice

1 Introduction

In movies, as in video games, modeling natural phenomena is becoming increasingly popular. In recent years, many such phenomena such as floods, earthquakes or even cast of lava, which were long impossible to recreate realistically on the screen, have been realized numerically. They could be seen especially in the 2004 movie "The day after tomorrow" or even more recently in the 2009 movie "2012".

Matter exists in different states, the most common being liquid, solid and gas. The dynamics of a particular state depend strongly on the previous state. For example, the form of ice formed by freezing depends on the dynamics of water right before the state change. The rules defining ice creation are based on fluid mechanics and thermodynamics. It should be possible, with a physical simulation, to create realistic looking ice. The use of a physical simulation often requires a very long computation time which does not leave much room for interactivity. Moreover, several non-intuitive parameters must be manipulated. When a particular appearance is desired, it is difficult, tedious and sometimes impossible to find the right values of these physical parameters.

A procedural ice model is proposed to create realistic glaze ice and icicles. It provides proper control, leaving plenty of room for creativity, and can be applied on complex geometric arrangements. Our most important contributions are:

- A reorganization of the simulation steps allowing greater control and a quick computation;
- A procedural model of icicle and glaze ice formation;
- An efficient computation of approximate water flow on a surface;
- An automatic positioning of the dripping water locations that create icicles;
- Collision handling of icicles;
- Interactivity on modeling of icicles.

J. Gagnon
Mokko Studio, R&D
Montréal, Canada
Tel.: (514) 932-4191
E-mail: jgagnon@mokkostudio.com

E. Paquette
Multimedia Lab
Département de génie logiciel et des TI
École de technologie supérieure
Montréal, Canada
E-mail: eric.paquette@etsmtl.ca

2 Related works

Frost is created when water is in the form of micro-droplets. The phase field has been used to simulate this type of solidification [5]. The approach also gives to the user some control by allowing the specification of a source image that will affect the appearance of the frost. To increase realism, a hybrid algorithm was created combining a phase field, a diffusion-limited aggregation and a fluid simulation [6]. These methods yield plausible frost creation results. However, as they manage only thin films of ice, it is impossible to create other shapes such as ice of sufficient thickness, or icicles. These techniques deal with ice crystals while glaze ice and icicles are different phenomena.

Glaze ice is created when water is in the form of drops. Metaballs were used to model static drops on a surface [13]. One of the constraints of this model is the loss of volume, which is resolved in the work of Tong et al. [11]. A spring system was used by Fournier et al. [2] for modeling water drops which provides a more realistic result. To simulate the motion of the drops, a particle simulation is used. However, this method does not support merge or split of the water drops. Wang et al. [12] solve this problem. They take into account the contact angle at the intersection between the liquid and the surface. The results are very accurate but they take nearly a week to compute. With these methods, it is possible to get the path of water drops on a surface; the methods are nevertheless more accurate than what is necessary for the creation of icicles and they do not allow calculations in interactive time.

What characterizes icicles is the solidification of water drops that flow from the surface to a point where they drip. To understand what is involved in creating an approach that addresses this type of phenomenon, theoretical models that exist in glaciology are first presented. There are several models for the growth of icicles. Makkonen et al. [7] propose an approach taking into account convection and conduction to model the outward horizontal growth, the inward horizontal growth and the vertical growth towards the tip. Maeno et al. [8] propose a model including dendritic growth at the tip of the icicle. The model by Szilder et al. [3] allows the prediction of the shape and mass of the icicle, as well as the frequency of water drops, as a function of time.

These models are essentially based on empirical observations made by their authors, and they do not provide methods to simulate the growth of icicles. However, they provide a comprehensive set of rules for understanding the mechanics of icicles' formation. The approach proposed in this work follows the empirical observations of the growth of icicles, giving plausible results without calculating the dynamics of each water drop.

Icicles were studied in computer graphics by Kharitonsky et al. [4] who propose an ice model based on thermodynamics using a discrete simulation. This model takes into account surface tension, contact angle, heat transfer, conductivity and the tendency of water drops to follow a path that is already wet. Although the result is realistic, this method cannot simulate several icicles at the same time. In addition, the calculation is done only on the icicle itself. Therefore, the result of the simulation must be attached manually on the surface of the object. Kim et al. [5] have tackled the problem of icicle formation by proposing a simulation based on the Stefan problem, which can simulate several icicles. Although the results of this work are realistic, they require a lot of computations. A simulation takes between 5 and 30 minutes and further, the user has no control over the positioning of the icicles and very limited control over the shape of the icicles.

The proposed approach greatly reduces the problem of computation time and lack of control over results. It automatically calculates the distribution of water on the surface and the points where the water drips from the surface. These calculations are done without a physically based simulation, which gives results in interactive time. Apart from providing a fast computation that allows for interactive modeling, the proposed approach offers several parameters that control various aspects of the position and shape of the icicles. Moreover, the approach can handle objects with complex surfaces such as trees.

3 Icicle definition

This section describes the general process behind the creation and growth of the icicles. By definition, an icicle is created when a stream of water (or other liquid) is in a configuration where it could drip from the surface of an object. Under the right conditions, the water still in contact with the surface may solidify [4].

An icicle grows by increasing its length and diameter [8]. When water flowing on the surface of the icicle leaves a film of water, it creates a new layer of ice that affects the diameter of the icicle [4]. This thin layer of water covers the surface of the icicle uniformly unless the amount of water is very small [7]. If there is enough water, it gets to the tip and only then can the icicle grow in length. There is a drop of water in suspension at the tip of the icicle, and it grows until it reaches a certain size before falling. The shape of this drop can be approximated by an hemisphere [7].

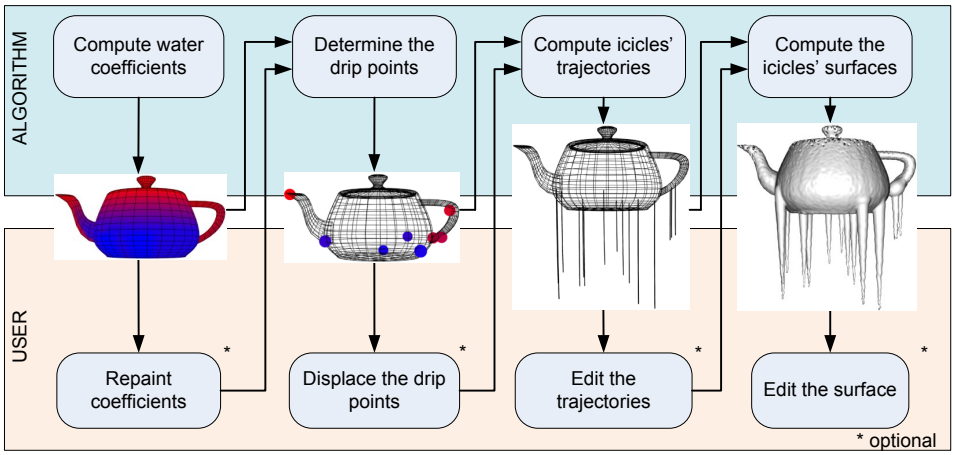

Fig. 1 Steps of the interactive icicles modeling

4 Procedural icicle modeling

The basis of the proposed method is to breaking down the icicle creation process into four stages. This decomposition allows for a fast computation, an increased control, and the possibility for the user to easily adjust the results of each stage. Although the results can be adjusted between the stages, user intervention is not required: the method can automatically generate realistic icicles from the input surface.

The first stage computes the flow of water on the input polygon mesh. It stores water coefficients at every vertex of the mesh. The second stage determines the regions where the water falls from the surface; this is where icicles are likely to grow. The third stage uses the water coefficients to create the trajectories along which the icicles grow. The fourth and last stage creates the surface of the icicles based on the trajectories. Various parameters provide control over the different aspects of the icicles creation. If more control is needed to achieve artistic goals, the result of each stage can be edited as shown in Figure 1; water coefficients can be painted, water drip points moved, icicle trajectories be edited, and the final mesh can be deformed.

4.1 Water coefficient computation

The first stage of the proposed method is the computation of the water flow. The goal is to determine, for each vertex, if the water reaches it and to compute an approximate amount of water. The water flow is used to select the locations where to grow the icicles and the amount of water is used to determine the growth rate of the icicles.

The distribution of water is governed by two factors: water reaches the surface providing the water supply, and then flows on the surface until it accumulates in a concave area or falls from the surface. When considering icicles, rain is one of the most important source of water [7]. In our implementation, the rain comes from a *source surface ss* provided by the user. Rain drops are computed using ray casting from the source surface according to the gravity vector. Ray origins are randomly distributed on the source surface based on the number of rays provided by the user. At each intersection point, upward facing vertices at a distance lower than an influence radius *rii* are added to the water supply. Visualization of the results is provided by assigning a red color to the vertices in the water supply and a white color to the others. Figure 2 presents an example of water supply in which the polygons are rendered by interpolating the colors of the vertices.

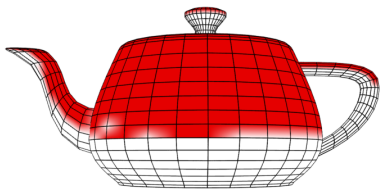

Fig. 2 Water supply on a teapot

With the defined water supply, the flow of water is simulated to compute the *water coefficients*. The water flow could be simulated using a particle system [1, 2], providing the detailed flow of each particle on the surface. In the proposed approach, the water flow is computed only from vertex to vertex along the edges. This approximation is sufficiently precise to create icicles and has the advantage of computing the flow very rapidly.

When computing the water flow along edges of the mesh, the physical process could be followed by moving

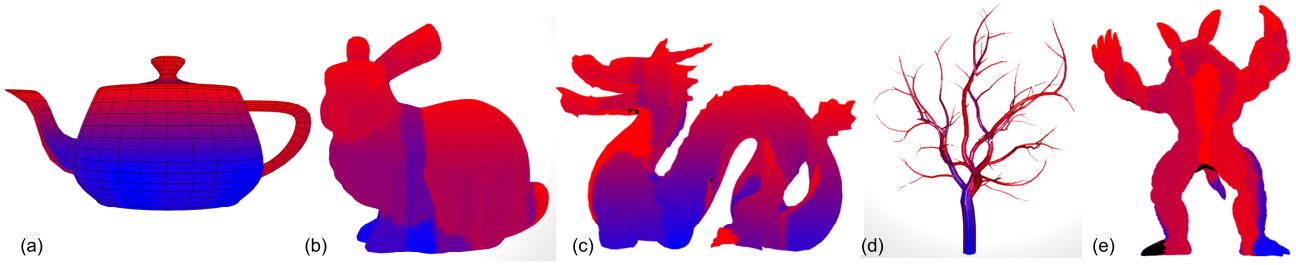

Fig. 3 Water coefficients on different objects

water from the vertices of the water supply downward until it reaches a dripping point. Since the proposed approach constrains the flow to follow the mesh edges, if every drop of water follows the most downward edge, it could leave some areas completely dry. To prevent this, water droplets would need to be split among several edges, greatly expanding the computation times. Furthermore, to compute ice glaze, the amount of water that flows on the surface must be known everywhere, so every simulated drop of water must adjust the water coefficient of every vertex it encounters. This results in several memory writes, which also reduce the efficiency of the approach, especially for a parallel implementation.

Because of these problems, another approach is proposed: computing the amount of water at a vertex by computing a path upward to the highest vertex that can be reached. Thus, the amount of water is written only once, when the highest vertex is found, and a parallel computation is efficient and easy to implement. Furthermore, every vertex that can reach some vertices in the water supply will get some amount of water, preventing the dry area problem of the downward approach. Even though this can have the opposite effect of spreading water to a larger area than what is realistic, it must be kept in mind that it is the icicles that are visualized, not the amount of water or the water flow.

The proposed approximate water flow computation meets the set goals: it provides satisfactory results, and can be computed very rapidly. The computation of the water flow is detailed in Algorithm 1. While computing the water coefficients, the goal is to get more water from larger surfaces (taken into account by term d in Algorithm 1) and to flow more water along edges that are oriented vertically (taken into account by term $-p$ in Algorithm 1).

If the user needs to edit the water coefficients wc, they can be visualized on the object mesh as a color code where red and blue vertices correspond respectively to lower and higher values of wc. Figure 3 shows the progressively varying values of the water coefficients that can be edited by the user when greater control is needed. With this approach, it is possible to efficiently compute water flow on most well formed meshes as can be seen in Figure 3.

foreach vertex v from the mesh **do**
  Current vertex $c = v$
  Water coefficient $wc = 0$
  while there are higher neighbor vertices to c **do**
    foreach higher neighbor vertex n **do**
      // Higher with respect to gravity **g**
      cn = normalized vector from c to n
      p = dot product (**cn**, **g**)
    Select neighbor $nmin$ for which p is minimal
    // The most upward n with respect to **g**
    if c or $nmin \in$ water supply **then**
      d = distance between c and $nmin$
      Multiply d by $-p$
      if only c or $nmin \in$ water supply **then**
        /* There is less water since one of the vertices is not in the water supply */
        Divide the result by 2
      $wc = wc +$ result
    $c = nmin$
  Save wc at vertex v

Algorithm 1: Water coefficients computation

4.2 Drip points identification

The result of the water flow from the previous stage provides an approximate quantity of water for each vertex. The water flow stops at two kinds of locations: concave regions where the water accumulates and convex regions where the water drips from the surface. To compute icicle creation and growth, we need to find the regions where the water drips. Generally, water moves according to the gravity like any other object and falls in the absence of a surface to hold it. Nevertheless, at the scale of water drops, it has the tendency to stick to the surface, even when the surface points downward. A water drop falls from the surface when the amount of water is sufficiently large and when the surface is sufficiently tilted. The region from which the drops can fall is shown in Figure 4.

The drip limit dl is set by the user as an angle with respect to the gravity vector. The polygons that form the *drip region* are those for which at least one of their vertices has a non-zero water coefficient, and for which the normal vectors of all their vertices satisfy the drip criterion (see Figure 4). The resulting drip region, as shown in Figure 5, contains the set of polygons from which the water is likely to drip. This drip region can

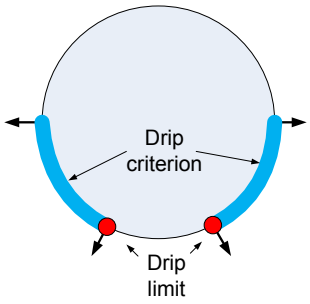

Fig. 4 Drip criterion

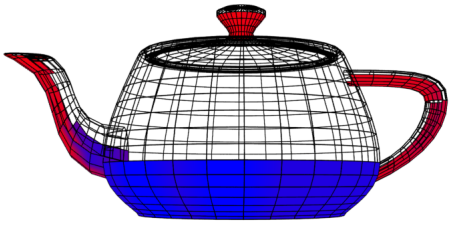

Fig. 5 Drip region

easily be edited by the user to remove areas where icicles should not be created. For example, it could be useless to create icicles at the bottom of objects that lie on top of others, such as the bottom of a teapot.

For objects with high curvatures modeled with few polygons, it should be possible to have areas where the drip criterion is realized, but where no polygon would satisfy the drip criterion for all of its vertices. In such cases, it is proposed to add the vertices and edges along the drip criterion transition to the drip region where icicles can be created.

Based on the number of icicles specified by the user, a set of points are randomly distributed on the drip region. These are the drip points, as shown in Figure 6 as circles with a color corresponding to the value of the water coefficient. The points and their associated water coefficients can be edited by the user if greater control is needed.

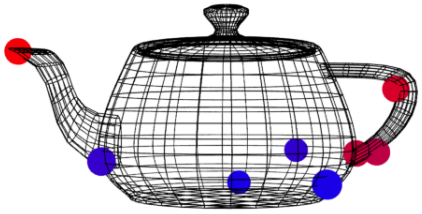

Fig. 6 Drip points

4.3 Icicles' trajectories definition

The third stage of the approach is the creation of the icicles' trajectories. A trajectory is created at each drip point and using the water coefficient, the final length of the icicle can be computed directly. This is another advantage provided by the proposed decomposition of the simulation in four stages. It bypasses the iterative simulation of the icicle growth, providing much shorter computation time, and enabling greater interactivity for the user.

In nature, the icicle growth depends on various properties, such as temperature, wind direction, amount of water, etc. This results in different types of icicles: linear, curved or ramified. The Icicles could also merge together. In the proposed approach, merging of icicles is handled at the surface creation phase (Section 4.4).

The trajectory computation stage proposes rules and control parameters that allow the creation of several types of icicles. In order to come up with appropriate rules, images of icicles were inspected. From this analysis, L-System [9] rules were developped to match the observations and provide intuitive control. The user can adjust the appearance of the icicles with a few parameters: curvature angle c, probability of subdivision d, and angle of dispersion a. The resulting L-System is as follows:

$$
\begin{array}{lll}
\omega & : FX & \\
p1' & : X & \xrightarrow{1-d} +(c)FFF/(a)X \\
p1'' & : X & \xrightarrow{d} [+(c)+(30°)F-(30°)FX]A \\
p2 & : A & \rightarrow +(c)FFF\backslash(a)X
\end{array}
$$

These rules are visually presented in Figure 7. Rule $p1'$ corresponds to the growth of the icicle while rule $p1''$ creates a new branch. Rule $p2$ is used to grow the main trunk after the creation of a branch. Parameter c represents the angle of curvature. While growing, the icicle is rotated by this angle at each step. Parameter a is also a rotation done at each growth step, but is a rotation (roll) around a trajectory. It affects the irregularity of the branch growth direction as well as the spread of the icicle branches. Parameter d represents the probability

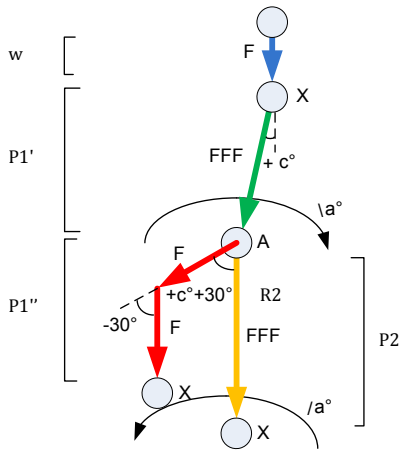

Fig. 7 Illustration of the L-System rules

of creating a new branch. The number of L-System generations is computed from the water coefficient, and can be adjusted by the user, should the resulting icicle be too long or too short.

The roll angle a is applied at each L-System generation. It is computed from the user specified angle au as $a = au \times 137.5°/360°$. The value 137.5° is often used when creating plants [9] to get a growth spiral with natural proportions. Thus, when $au = 360°$, the icicle branches spread uniformly. For lower values of au, the branches grow in a constrained direction. Figure 8 shows, from a top view of the icicles, the impact of different values of au on the spread of the branches.

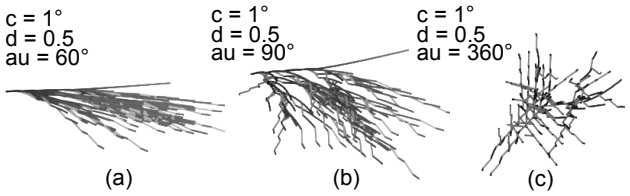

Fig. 8 Results with varying angles of dispersion

The curvature angle c is applied at each L-System generation. For linear icicles, c should have a low value. (see Figure 9(a)). When $c = 0$, the icicle follows a perfect straight line. Windy conditions often result in curved icicles that can be created by adjusting the value of c, such as in Figure 9(b) and (c). All of the different parameters provide adequate control and enable a broad range of icicle shapes as seen in Figure 10. Each related trajectory is composed of line segments generated by the L-System. It will guide the creation of the icicle surface. The user can adjust the shape of the trajectories with various parameters, and the new trajectories are generated almost instantly, providing a great

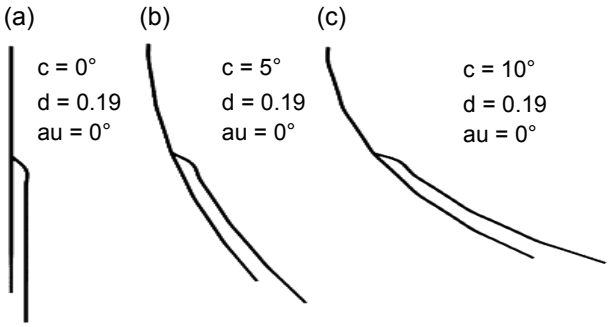

Fig. 9 Curved trajectories

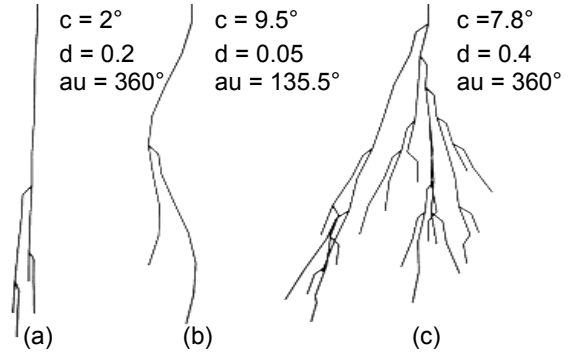

Fig. 10 Ramified trajectories

level of interactivity. The possibility of visualizing the trajectories interactively is a major advantage versus physical simulation approaches, in which long simulation times and poor control over the results hinder the artistic creativity of the user.

4.3.1 Collision detection handling

While growing, icicles can collide with the surface of objects. In such cases, the icicle follows the surface in the direction of the gravity until it reaches a drip region, and then, the trajectory continues to grow from the drip region entry point. As seen in Figure 11, several surfaces can interfere with the growth of the icicle, but it will still produce a realistic result.

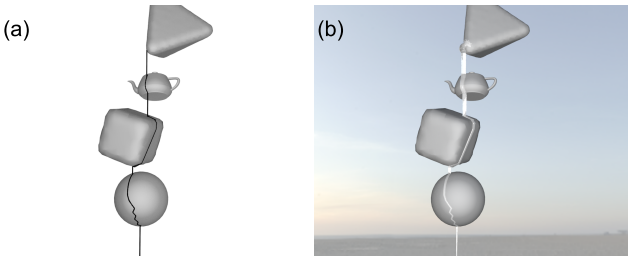

Fig. 11 Icicle trajectory that intersects with several objects

4.4 Surface creation

The last stage of the proposed approach is the creation of the surface. The surface is created around the trajectory from the previous stage by computing the varying radius along the trajectory. In order to do this, a profile function was developped.

4.4.1 Icicle profile

The creation of the profile function was inspired by models from the glaciology field. The experiments by Maneo et al. [8] showed that the radius of the drop at the end of the icicle is 2.44 millimeters on average. Icicles grow 10 to 30 times faster along their length than along their diameter [7], which results in a tube like shape. As water flows on the surface, it leaves a thin film along the icicle. If this water film freezes, it increases the diameter of the icicle [4]. Depending on how fast water freezes on its way down the icicle, the resulting shape will look more like a cylinder or a cone. Icicles can also look more or less wavy, mostly depending on the turbulence of the wind field. To model all these aspects, three parameters are provided to the user: as and fs which respectively control the amplitude and the frequency of the undulation, and t, which controls the conical shape. The profile function uses these parameters to compute the radius of the icicle along the trajectory:

$$R(x) = tip + (L - x) \times t + as \times \sin(x \times fs) \quad (1)$$

For a given position x, from the base of the icicle ($x = 0$) to the tip ($x = L$), $R(x)$ provides the radius of the icicle. In Equation 1, tip is the radius of the tip of the icicle. It is adjusted to a value equivalent to 2.44 millimeters and an hemisphere of this size is used to model the surface. Going upward from the tip, the radius grows as a function of $(L-x)\times t$. This term controls the size of the base of the icicle. Finally, the term $as \times \sin(x \times fs)$ controls the undulation of the icicle.

4.4.2 Modeling

With the profile function, it is now possible to generate a surface. The surface is constructed through the use of metaballs, which also allow the fusion of several icicles. Metaballs are positioned on the points of the trajectory, and their radii are derived from the profile fonction. As illustrated in Figure 12 (a)-(c), it is possible to generate surfaces that are somewhat conical and undulated.

The approach proposed by Kim et al. [5], uses a sine wave to simulate the ripples on the surface. The application of a sinusoidal function provides a very regular surface. Indeed, the ripples of the icicle are symmetrical which is not very realistic. To make the model more plausible, the proposed approach allows to add a noise nc, not at rendering time as is the case of Kim et al. [5], but during the construction of the surface. Rather than positioning metaballs only along the trajectory, they are positioned randomly, within a distance nc, in the plane perpendicular to the trajectory. As illustrated in Figure 12(d)-(f), the results are less uniform and much more realistic. Furthermore, the global shape, undulation, and noise of the results can be adjusted to correspond very well to what can be observed on photographs of real icicles (Figure 12(g)).

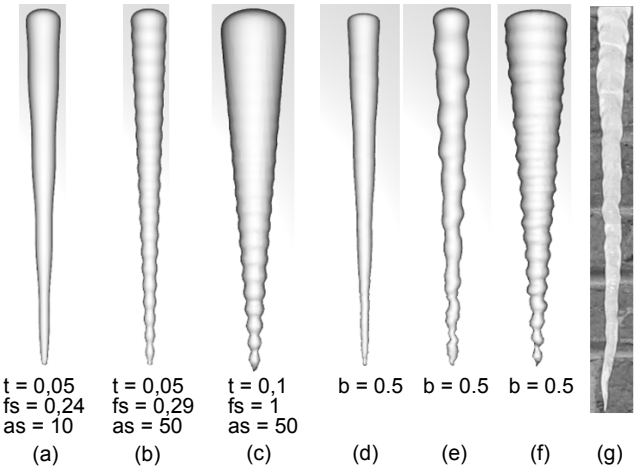

Fig. 12 Control of the icicles' surface

4.4.3 Base of the icicle

The base of the icicle is where the icicle attaches itself to the surface of the object. Its shape is quite variable, and is influenced by the water supply amount. In nature, it can be observed that the thickness of ice on the surface increases considerably the closer it gets to the drip point. To create the area of the base of the icicle, a set of metaballs is distributed over the surface of the object around the drip point, up to a distance eb specified by the user. The number of metaballs distributed on the surface is also specified by the user and is noted nmb. The radius of each metaball is calculated as follows:

$$rb = \frac{(eb - d)^2}{eb^2} \times wc \quad (2)$$

where d is the distance between the position of the drip point and the current metaball and where wc is the water coefficient. The result, as illustrated in Figure 13, is an icicle base that adapts well to various surfaces such as (a) a sphere, (b) a rotated cube, and (c) a teapot.

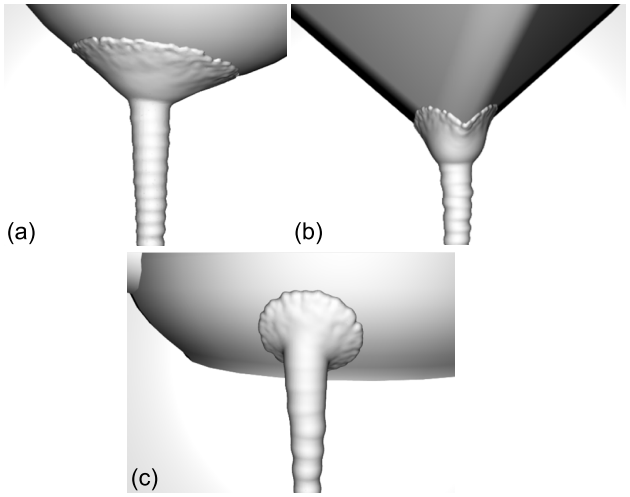

Fig. 13 Base of icicles on different surfaces

4.4.4 Glaze ice

Glaze ice is an ice layer which covers the initial surface, as shown in Figure 14. It is possible to observe three types of solidification: water freezes quickly, Figure 14 (a), uniformly, Figure 14 (b), or slowly, Figure 14 (c).

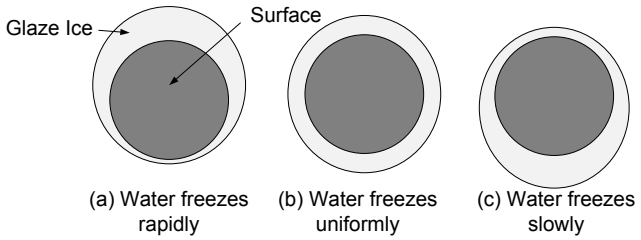

Fig. 14 Different types of glaze ice

The glaze ice is computed using the water coefficient and a coefficient called water drop lifetime, noted k. Metaballs are distributed randomly on the surface. The number of metaballs is a user-defined parameter noted ngi. To generate the different glace ice types, the radius of each metaball is computed, as in Equation 3

$$rgi = minGI + s \times [dUp \times lt + dDown \times (1 - lt)] \quad (3)$$

where rgi is the resulting radius, dUp is the distance between the current vertex and the highest vertex, $dDown$ is the distance between the current vertex and the lowest vertex in the drip region, $minGI$ is the minimum thickness of the glaze ice and s is the scaling of the glaze ice thickness. The distance dUp and dDown are computed as in the water flow simulation. As shown in Figure 15, when the value of lt is equal to 0, the first type of glaze ice is achieved; 0.5 provides the second, and 1 the third.

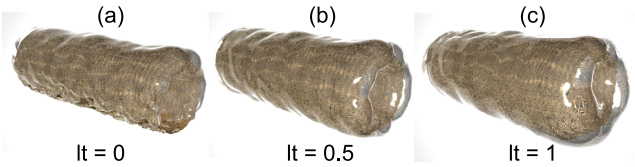

Fig. 15 Results of different types of glaze ice

5 Results

Table 1 Control parameters

	Description	Domain	Typical value
Water coefficient			
ss	Source surface	-	-
g	Gravity vector	-	(0 -1 0)
nr	Number of rays	]1..∞[	500
rii	Radius of influence at the intersection	]0..∞[	1
Drop points			
ns	Number of icicles	]1..∞[	10
dl	Drip limit	]1..90°[	75°
Trajectories			
c	Curve angle	[0..360°[	2°
d	Division probability	[0..1[	0.1
au	Roll angle of growth and dispersal	[0..360°]	360°
Surface modeling			
t	Growth ratio	[0..∞[	0.5
as	Sine wave amplitude	[0..∞[	0.3
fs	Sine wave frequency	[0..∞[	50
eb	Base extent	[0..∞[	1
nmb	Number of metaballs at the base	[1..∞[	100
nc	Noise coefficient	[0..∞[	1
Glaze ice			
minGI	Minimum radius for the glaze ice	[0..∞[	0
ngi	Number of metaballs used for the glaze ice	[0..∞[	5000
s	Scaling of the glaze ice	[0..∞[	1
lt	Lifetime a Water drop	[0..1]	0.5

The proposed approach can successfully create icicles and glaze ice on various complex surfaces. It allows proper control over the look of the final result. Although this control can be done using several parameters (Table 1), the user typically manipulates only a few of them. For example, only four parameters ns, c, d and fs must be adjusted to produce all the results presented in this section. Tests were conducted on several different surfaces. The results are shown in Figure 16 and the computations are detailed in Table 2. The results were computed on an Intel Xeon 3.4 GHz Dual-core. Objects from Figures 16(c)-(f) are a courtesy of the Stanford University Computer Graphics Laboratory. As can be

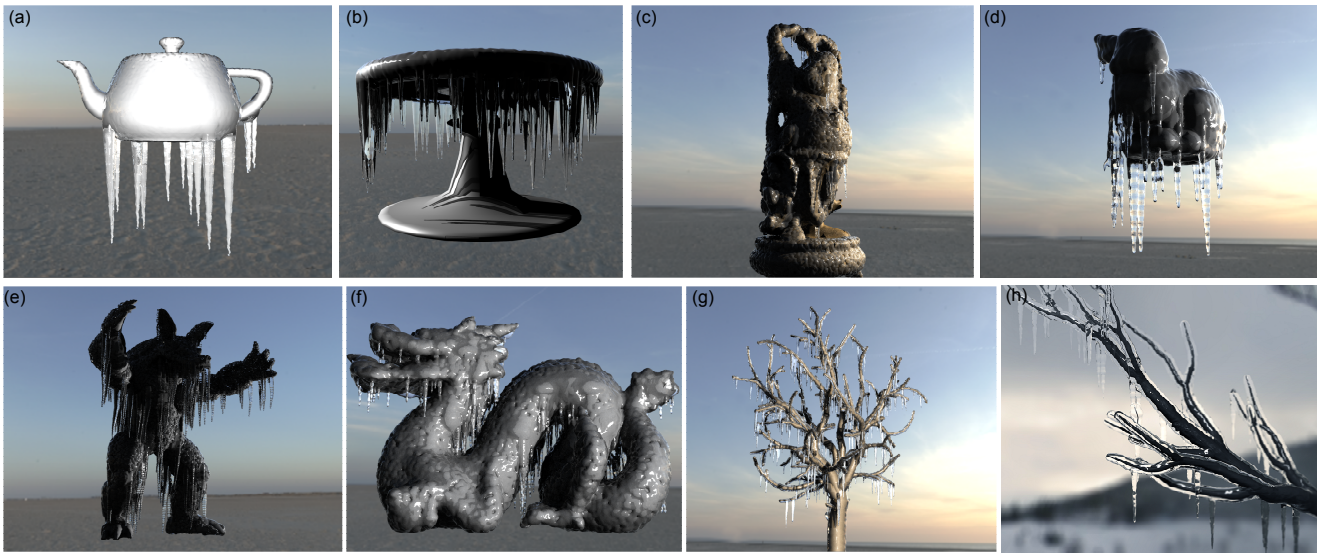

Fig. 16 Procedural icicle modeling on various objects. Rendered images are courtesy of Mokko Studio.

Table 2 Statistics for the results of Figure 16

Figure 16	(a)	(b)	(c)	(d)	(e)	(f)	(g)
Input object number of polygons	1,617	2,240	15,536	3,851	30,780	11,102	6,660
Number of icicles	12	300	10	20	200	200	400
Icicles number of polygons	10,400	179,428	749,296	102,594	390,616	244,188	131,000
Computation times (seconds)							
Water coefficients	0.1	0.1	0.5	0.3	5	0.4	0.3
Drip points	0.1	0.5	0.2	0.1	0.5	0.5	0.5
Trajectories	0.1	0.2	0.1	0.2	0.5	0.5	1
Surface modeling	11	55	60	20	116	19.5	36
Total time	11.3	55.8	60.8	20.6	122	20.9	37.8

seen, the proposed approach is relatively fast as compared to previous work. Indeed, by way of comparison, the icicles found on the fountain of the paper by Kim et al. [5] (Figure 5 in their paper) took 30 minutes to simulate (on a 3 GHz Pentium 4). While the proposed method is not physically based, it provides lots of control to the user, making it easy to create of a fountain with very similar icicles (Figure 16(b)) computed within only 55 seconds (on a Xeon 3.4 GHz Dual-core). When comparing to the work by Kharitonsky et al. [4], the proposed method generates icicles with the same visual quality, while adding the automatic and realistic positioning of the icicles and providing a convincing ice surface that attaches to the object surface. It is therefore possible to model hundreds of icicles on much more complex surfaces, in a reasonable amount of time.

5.1 Limitations

While the water flow phase can handle most well formed meshes, some objects pose further challenges. Firstly, objects with disconnected or intersecting polygons sets do not allow for the computation of the flow solely along the edges. Secondly, highly detailed objects are most likely to contain several polygons that are superfluous to the water flow computation. This detailed polygon set results in a more expensive flow computation and can result in several drip locations where interesting icicles are unlikely to grow. To compute water flow on these problematic object types, an approximate surface should be constructed to allow the water flow computation. Creating the approximate surface using implicit surfaces should smooth out the problematic areas and provide a surface adequate for the proposed water flow.

To obtain a fast computation time for interactive use, some compromises on the accuracy of the solidification phenomenon have been made. Our approach does not use fluid simulation equations to calculate the motion of water on objects. Rather it is a model that is realistic without being based on a physical simulation, and the output result cannot be seen as a physically accurate representation. The proposed approach is in fact a compromise between fidelity, aesthetics, efficiency and control.

6 Conclusion

The proposed approach demonstrates the power of the decomposition of a complex problem, such as simulating the formation of icicles, in several interactive stages. In the proposed approach, the freezing process is divided into four stages. The first stage determines how the water flows on the object and it is composed of two sub-stages: definition of the water supply and traversing of the surface by calculating the water coefficients. The proposed approach efficiently computes the water flow, and the calculation can be easily parallelized.

The second stage determines the drip points and it is composed of two sub-stages: identification of drip regions and positioning of drip points. This stage provides good control over the icicles positions.

The third stage generates the trajectories of icicles, and an L-System has been defined which allows a wide variety of ice growth. The user can change the curvature, the probability of division, and dispersal of branches. In addition, collision detection between the trajectory and the objects is supported.

The fourth stage is the surface construction. A surface is built around each trajectory. The radius of the surface is defined using a profile function that enables the ripple and the conical shape of the surface to be adjusted. A random perturbation is used on the positioning of metaballs to create realistic ripples. Metaballs are also used to manage the fusion of several icicles. Finally, the object can be coated with different types of glaze ice using an equation which manages the thickness of ice on the surface.

The advantage of this decomposition into stages is the control and the interactivity it allows: the user has access to several parameters and can see their results in seconds. The total computation time is considerably faster than seen in previous works. The result is a photorealistic icicle modeling similar to results of physically-based simulations.

6.1 Future work

The proposed approach is fast, but it is not instantaneous. The user can have a good idea of the final result from the trajectories which are computed within seconds. Nevertheless, it takes much more time to compute the final geometry. An approach to solve this problem would be to provide an appropriate preview of the surface without the use of the time-consuming metaballs. The computation of the final geometry should also be accelerated. Instead of using metaballs, the direct computation of the icicles' mesh should be investigated.

Creating the glaze ice surface is also quite time-consuming. At the same time, as mentioned in Section 5.1, an alternative surface is sometimes required to compute the water flow. A unified approach adapted from surface reconstruction methods should be developed, with the computation times and surface smoothness constraints in mind.

Acknowledgements The authors would like to thank the Biological Modeling and Visualization Research Group, Przemek Prusinkiewicz and Steve Longay from the University of Calgery for providing us with several tree models used to validate our approach. We would like to thank the artists of Mokko Studio for their constructive comments. Thanks also to Nicolas Rous and François Dagenais from the Mokko research laboratory for discussions and suggestions, and finally, Danny Bergeron, president of Mokko Studio, for allowing us to undertake this project. This research fits in the Animation, Graphics, and Imaging theme of the GRAND Networks of Centres of Excellence.

References

1. Dorsey, J., Pedersen, H.K., Hanrahan, P.: Flow and changes in appearance. In: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, SIGGRAPH '96, pp. 411–420. ACM, New York, NY, USA (1996)
2. Fournier, P., Habibi, A., Poulin, P.: Simulating the flow of liquid droplets. Proc. Graphics Interface 98 (1998)
3. K., S., E., L.: An analytical model of icicle growth. Annals of Glaciology **19** (1994)
4. Kharitonsky, D., Gonczarowski, J.: A physically based model for icicle growth. Vis. Comput. **10**, 88–100 (1993). DOI 10.1007/BF01901945
5. Kim, T., Adalsteinsson, D., Lin, M.C.: Modeling ice dynamics as a thin-film stefan problem. In: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation, SCA '06, pp. 167–176. Eurographics Association, Aire-la-Ville, Switzerland (2006)
6. Kim, T., Henson, M., Lin, M.C.: A hybrid algorithm for modeling ice formation (2004)
7. Makkonen, L.: A model of icicle growth. Journal of Glaciology **34**(116) (1988)
8. N., M., L., M., K., N., K., K., T., T.: Growth rates of icicles. Journal of Glaciology **40**, 319–326 (1994)
9. Prusinkiewicz, P., Lindenmayer, A.: The algorithmic beauty of plants. Springer-Verlag pp. 101–107 (1990)
10. Szu-Han Chen, A., Morris, S.W.: Experiements on the morphology of icicles. Physical Review E (2010)
11. Tong, R., Kaneda, K., Yamashita, H.: A volume-preserving approach for modeling and animating water flows generated by metaballs. The Visual Computer **18**(8), 469–480 (2002)
12. Wang, H., Mucha, P.J., Turk, G.: Water drops on surfaces (2005)
13. Yu, Y.J., Jung, H.Y., Cho, H.G.: A new water droplet model using metaball in the gravitational field. Computers and Graphics **23**(2), 213–222 (1999)