Eurographics/SIGGRAPH Symposium on Computer Animation (2003)
D. Breen, M. Lin (Editors)

Visual Simulation of Ice Crystal Growth

Theodore Kim and Ming C. Lin

Department of Computer Science, University of North Carolina at Chapel Hill, U.S.A.
http://gamma.cs.unc.edu/ICE

Abstract

The beautiful, branching structure of ice is one of the most striking visual phenomena of the winter landscape. Yet there is little study about modeling this effect in computer graphics. In this paper, we present a novel approach for visual simulation of ice growth. We use a numerical simulation technique from computational physics, the "phase field method," and modify it to allow aesthetic manipulation of ice crystal growth. We present acceleration techniques to achieve interactive simulation performance, as well as a novel geometric sharpening algorithm that removes some of the smoothing artifacts from the implicit representation. We have successfully applied this approach to generate ice crystal growth on 3D object surfaces in several scenes.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism

1. Introduction

The geometrically and optically complex structure of ice is one of the most striking visual phenomena in winter. These beautiful, branching patterns of ice can be found on many exposed surfaces, such as sidewalks, panes of glass, and hoods of cars. Together, these surfaces comprise a unique aspect of the frozen, wintery landscape.

While there exists some work that models this visual complexity [26, 28], there has been relatively little research in computer graphics that attempts to physically simulate the growth of ice patterns. In addition, none of the previous work presents a mechanism that allows an artist to automatically adjust the simulation parameters to achieve a specific visual effect. However, there is a large body of knowledge in both crystal growth and computational physics that addresses the computation of the liquid to solid phase transition. There exists a wide morphology of ice patterns, and in this paper we present a model that can simulate several different types of solidification, most notably *dendritic* solidification, which is the most geometrically complex and visually interesting of all ice structures.

Main Contributions: We present a novel approach for the visual simulation and rendering of ice crystal growth. We choose a simple and powerful simulation technique from the crystal growth literature, known as the *phase field* method. We present techniques to simplify the computation and make the problem of simulating modest-scale dendritic ice crystal growth more tractable. We also show how the phase field method allows a user parameterization that a visual effects artist can use to manipulate the ice crystal growth. The phase field method often has smoothing artifacts as a result of its implicit representation, and it can only compute the outermost ice-water boundary. Therefore, a novel intermediate geometric processing step is introduced to add sharp edges and medial ridges to the interior of the ice. Finally, the simulated images are rendered using photon mapping [14]. In comparison to the existing work in this area, our method offers the following advantages:

- Physically-based ice growth based on rigorous mathematical formulations and sound physical observations;
- Simple and natural aesthetic control of simulation parameters for generating desired visual effects;
- A physically-inspired, novel geometric processing step that introduces internal structure to the ice and enhances the visual realism of the final rendered image;
- Accelerated and simplified computations for interactive simulation of modest-scale ice crystal growth.

The basic simulation and rendering framework has been applied to several different scenarios. Fig. 1 shows an example image generated by our method.

Organization: The rest of the paper is organized as follows. A brief survey of related work is presented in section 2. Section 3 gives an overview of our approach. We present the numerical method and acceleration techniques we used to simulate the ice growth in section 4. Section 5 introduces our aesthetic control parameters, which can be used to drive the simulation toward intended visual effects. We describe a novel geometric method to create crests and ridges in the ice

Figure 1: ***Detail of ice grown on a stained glass window.*** The inset shows the full window.

in section 6. The results of our implementation are shown in section 7.

2. Previous Work

In this section, we briefly survey related simulation and rendering techniques from the computer graphics and computational physics literature.

2.1. Visual Simulation Methods for Water in Different States

The visual simulation and modeling of the other states of H_2O have been well-studied in the past. The dynamics of water and steam have been impressively captured in general fluid simulations [7, 9]. Recently, Fearing[8] examined the solid state of water when simulating the dynamics of fallen snow. However, the analysis in his paper focused on the phenomena that arise in deposition and drift of snow, not those that arise in the liquid to solid phase transition. Instead, it assumed that all phase transitions had already taken place in the sky. Consequently, the lack of ice in Fearing's scenes is noticeable.

A famous empirical algorithm that attempts to capture the structure of dendritic ice is the Koch snowflake. First described by Helge von Koch in 1904 [26], it defines simple production rules that, when applied recursively, produces a structure that is in close visual agreement with that of a snowflake. The reader is referred to *The Fractal Geometry of Nature*[17] for further details. While it renders visually plausible results, the Koch snowflake has no clear physical basis and certainly does not allow for an aesthetic parameterization.

Diffusion Limited Aggregation, or DLA,[2, 28] is a technique from physics that attempts to capture similar effects to the ones we describe here. Notably, this technique deals with solidification in the context of *vapor deposition*, the aggregation of water molecules in the air onto a cold surface. Instead, we present a method that models the aggregation of liquid molecules on a crystal in an undercooled melt.

2.2. Simulation Techniques in Computational Physics

Ice can take many geometric forms, from the uninteresting structure of ice cubes to the dendritic growth we examine in this paper. For an introduction to the morphology of possible ice crystal shapes, the reader is referred to the paper by Yokoyama and Kuroda[29].

In addition to the visual appeal of dendritic crystals, their simulation is also of considerable practical interest. During the creation of alloys, a liquid to solid phase transition occurs, and if an dendrite forms in the melt during this process, the alloy can be drastically weakened. Consequently, there is a considerable body of work in the computational physics and crystal growth literature addressing this problem, and one of the most interesting simulation techniques that has emerged is the phase field method [16].

In its simplest form, the phase field method can be very computationally expensive. Therefore, various acceleration techniques have recently been developed to make the computation more tractable. These include adaptive mesh refinement [20] and diffusion Monte Carlo [19] techniques. We will instead propose both a simpler scheme and a mapping to graphics hardware. Both techniques accelerate simulation performance and make it suitable for modeling modest scale ice growth.

At its core, the problem of dendritic solidification is one of tracking an evolving interface. Thus, the level set method [18, 22], an approach that has been widely used in computer graphics recently, can also be applied to the problem. While traditionally there have been problems in the use of level set methods to simulate dendritic solidification, many of them have been addressed in recent work [10]. The level set method is also capable of providing a solution of higher order accuracy than the phase field method. However, we feel that this level of precision is unnecessary. Both the phase field and the level set methods can support an aesthetic parameterization, but we have chosen to use phase fields because it is simpler to implement and optimize, particularly on graphics hardware. Notably, the level set method is also an implicit simulation technique, and suffers from the same smoothing artifacts as the phase field method. Consequently, if level set methods were used in place of the phase field method, our geometric sharpening step would still be necessary.

3. Overview

We give a brief overview of the overall computational framework and the basic design of each step involved.

We choose a simple and powerful implicit simulation technique from the crystal growth literature, known as the *phase field* method. This method can take $O(N^3)$ time, where N is the resolution of a single grid dimension. To obtain reasonable accuracy, N must be fairly large, making the computation quite expensive. We reduce the computation time significantly by using two acceleration techniques. The first is based on the observation that most ice crystals are very thin. We can simulate growth in 2D and add 3D detail later,

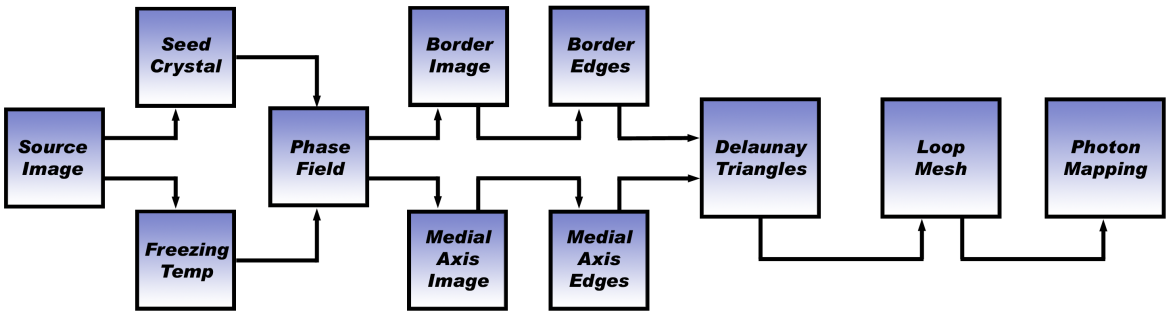


Figure 2: The overall system pipeline.

reducing the computation time from $O(N^3)$ to $O(N^2)$. Second, we further improve the performance of the simulation by performing banded computation around the "front" of the ice and water interface, instead of over the entire grid.

We then adapt the phase field method to include aesthetic controls for a visual effects artist to manipulate. This is achieved by user control of the seed crystal and freezing temperatures input into the phase field simulation.

For the seed crystal, we use the visually salient features of our target object. The features are extracted using edge detection and used to set the initial conditions of the simulation. In addition to seeding the simulation, we also influence the simulation throughout by manipulating the freezing temperature.

Due to the smoothing artifacts of the phase field method and the lack of internal detail given by the evolving interface, a novel intermediate geometric processing step is introduced to add sharp features prior to rendering. This is performed by first computing the border and medial axis of the ice with morphological operators. Given the resulting medial axis and boundary edges, we generate a constrained conforming Delaunay triangulation upon which a subdivision step is performed to introduce creases and edges [6]. Finally, the triangles are rendered using photon mapping.

Fig. 2 shows the overall system pipeline of our computational framework. Next, we will describe each step in greater detail.

4. The Phase Field Method

In this section, we describe the *phase field* method, a numerical technique used to simulate undercooled ice growth. Subsections [4.1] - [4.3] give an overview of the method, and present Kobayashi's formulation [16]. In subsections [4.4] - [4.7] we will present our own analysis and optimizations.

4.1. Undercooled Solidification

An undercooled liquid is a liquid that has been cooled below its freezing temperature, but has been cooled sufficiently slowly for it to remain in its liquid state. When a small amount of solid material, known as the seed crystal, enters a container filled with undercooled liquid, the liquid transitions to solid radially outwards from the initial seed in a rapid and unstable reaction. Due to this instability, the growth of the crystal can be influenced by small perturbations, such as surface tension or minute impurities in the liquid. These factors can lead to the complex branching, or "dendritic", behavior we see in ice.

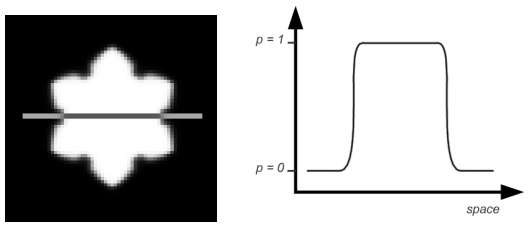
4.2. The Phase Field

In the phase field method, the undercooled liquid is represented implicitly as a two or three-dimensional grid. This is also known as an 'Eulerian' representation. Several graphics papers describe Eulerian simulation in detail [11, 27], as does any general applied linear algebra text [5]. For simplicity and tractability, we limit our simulations to two dimensions.

Two separate fields are tracked using this discrete representation: A temperature field T, records the amount of heat in a given cell, and a phase field p records the current phase of a given cell. For a given grid coordinate (x,y), we define T_{xy} and p_{xy} as the corresponding values in the temperature and phase fields.

For a given (x,y), if $p_{xy} = 0$, the cell is filled with water, and if $p_{xy} = 1$, the cell contains ice. If p_{xy} is between $[0,1]$, then it is at an intermediate stage between the two states. While we usually think of phase as a binary state, either water or ice, on the microscopic level there is a continuum of states along the ice front. The phase field method makes the computation of solidification tractable by magnifying this microscopic continuum so that it is visible macroscopically.

Fig. 3(a) is an example of a partially reacted phase field, and Fig. 3(b) is a cross section from Fig. 3(a). The horizontal axis of Fig. 3(b) is the spatial dimension, and the vertical axis is the phase dimension. In actuality, the transition from $p = 1$ (ice) to $p = 0$ (water) should be a microscopically thin, virtually instantaneous step function. Instead, the microscopic transition has been magnified, and we can see a region of quick but finite transition. Once the interface has been magnified to a resolution where non-integral values of p_{xy} appear on the grid, we can evolve the interface by applying a pair of partial differential equations.



(a) (b)

Figure 3: (a) A phase field in which white is $p = 1$ (ice), and black is $p = 0$ (water). The gray band in the middle is the section shown in profile in (b). (b) Cross section from (a) in profile. Note that while the transition from water to ice is abrupt, it is not instantaneous.

4.3. The Kobayashi Formulation

The first paper to report successful simulation of a wide variety of ice growth patterns using phase fields is by Kobayashi [16]. His formulation is similar to the reaction-diffusion equations [27, 25] for texture synthesis in computer graphics. In reaction-diffusion, the propagation of chemicals through a medium is described using a pair of PDEs of the form:

$$\frac{\partial C}{\partial t} = a^2 \nabla^2 C + R.$$

On the right hand side, $a^2\nabla^2 C$ represents diffusion, and R represents an arbitrary reaction function. The a^2 is a spatially-variant anisotropy term. The PDE for Kobayashi's temperature field fits this form:

$$\frac{\partial T}{\partial t} = a^2 \nabla^2 T + K\frac{\partial p}{\partial t}. \tag{1}$$

The diffusion term remains the same, since we are in fact simulating heat diffusion. In this case, $R = K\frac{\partial p}{\partial t}$, where K is a latent heat constant. This R term models the process where, as water transitions to ice, it produces heat.

The phase field term in Kobayashi's formulation is significantly more complex than the previous equations:

$$\tau\frac{\partial p}{\partial t} = \nabla\cdot(\varepsilon^2\nabla p) - \frac{\partial}{\partial x}\left(\varepsilon\frac{\partial \varepsilon}{\partial \theta}\frac{\partial p}{\partial y}\right) + \frac{\partial}{\partial y}\left(\varepsilon\frac{\partial \varepsilon}{\partial \theta}\frac{\partial p}{\partial x}\right) + p(1-p)\left(p - \tfrac{1}{2} + m(T)\right). \tag{2}$$

The first portion is a diffusion term:

$$\nabla\cdot(\varepsilon^2\nabla p) - \frac{\partial}{\partial x}\left(\varepsilon\frac{\partial \varepsilon}{\partial \theta}\frac{\partial p}{\partial y}\right) + \frac{\partial}{\partial y}\left(\varepsilon\frac{\partial \varepsilon}{\partial \theta}\frac{\partial p}{\partial x}\right)$$

that is significantly more complex than the standard Laplacian. The standard Laplacian diffusion term ($\nabla^2 C$) is the sum of the diagonal elements of the *Hessian* matrix:

$$\begin{bmatrix} \frac{\partial^2 p}{\partial x^2} & \frac{\partial^2 p}{\partial x \partial y} \\ \frac{\partial^2 p}{\partial y \partial x} & \frac{\partial^2 p}{\partial y^2} \end{bmatrix}. \tag{3}$$

The Kobayashi diffusion term is also the sum of elements from the Hessian, but it takes into account all the matrix entries. The placement of the anisotropy term is also different, between the first and second partials. As a result, the diagonal terms are abbreviated as a gradient and divergence operator ($\nabla\cdot(\varepsilon^2\nabla p)$) instead of a pure Laplacian. This difference is significant, because $\nabla\cdot(\varepsilon^2\nabla p) = \varepsilon^2\nabla^2 p + \nabla\varepsilon^2\cdot\nabla p$. As a result, this different anisotropy placement accounts for both the Laplacian of the phase term and the gradient of the anisotropy term.

Kobayashi also presents a complex and general model of anisotropy. First, we define θ as the orientation of the front at a given grid cell, $\theta = -\nabla p$. In two dimensions, this reduces to $\theta = -\cos^{-1}(\frac{\frac{\partial p}{\partial x}}{|\nabla p|})$. The anisotropy term is then:

$$\varepsilon(\theta) = \bar{\varepsilon}(1 + \delta\cos(j(\theta_0 - \theta)) \tag{4}$$

where $\bar{\varepsilon}$, δ, j, and θ_0 are constants. The constant j is the degree of anisotropy, which defines preferred directions of growth. δ is the strength of anisotropy, which defines the speed of growth in the preferred directions. θ_0 is a fixed reference direction, and $\bar{\varepsilon}$ is the scaling factor that determines how much the microscopic front is magnified. The values we used for these and other constants is given in Table 1. The $\frac{\partial \varepsilon}{\partial \theta}$ term is also necessary in Eqn. 2, but this can be obtained by taking the analytical derivative of Eqn. 4.

α	γ	T_e	j	θ_0	$\bar{\varepsilon}$	τ	a
0.9	10.0	1.0	4.0	$\frac{\pi}{2}$	0.01	0.0003	1.0

Table 1: Simulation Constants. **Top:** Equation symbols; **Bottom:** Values used

Next we examine the reaction term in Eqn. 2.

$$p(1-p)\left(p - \frac{1}{2} + m(T)\right), \tag{5}$$

where the m term is defined as:

$$m(T) = \frac{\alpha}{\pi}\arctan(\gamma(T_e - T))). \tag{6}$$

Eqn. 5 models the energy potentials in the system. The details of this equation are probably of limited use to a graphics audience, so we will instead present some basic intuition.

When $m = 0$, Eqn. 5 is positive over $0.5 < p < 1$ and negative between $0 < p < 0.5$. So, the energy is in a "metastable" state where values of p are encouraged to stay the same. Conversely, when $m = 0.5$, Eqn. 5 is positive for all $0 < p < 1$. So, if a grid cell has $m = 0.5$, no matter what its p value, it is encouraged to transition towards ice. As the temperature of a grid cell increases, its m increases towards 0.5, and it becomes more likely to transition to ice.

Despite the complexity of the above discussion, Eqns. 1 and 2 are all that are necessary to simulate ice growth. We will not present a method of synthesizing ice onto 3D objects here, because the 2D texture synthesis methods presented by Witkin et al. [27] and Turk [25] can both be applied without modification.

4.4. Improved Anisotropy

Eqn. 4 affords both simpler and richer controls for general texture synthesis than the anisotropy term described by Witkin and Kass [27]. Witkin and Kass' formulation limits the number of preferred growth directions to 0, 2, or 4, and all the directions must be either parallel or orthogonal. Additionally, the strength of anisotropy in parallel directions must be the same. For example, if we prefer fast growth along the x axis, we cannot specify different speeds for the positive and negative directions.

Using the constant j in Eqn. 4, we can specify an arbitrarily high degree of anisotropy, and with a slight modification, specify a different speed for each direction. This is accomplished by defining a separate δ_i for each i^{th} cosine lobe, and limiting the influence of δ_i to the range $\frac{i*2\pi}{j} \leq \theta < \frac{(i+1)*2\pi}{j}$.

4.5. Possible Ice Crystal Shapes

In Fig. 4, we show the results of our simulation, starting from a point source of ice in the center. By varying the K and δ simulation parameters, we can produce the "isotropic", "sectored plate", and "dendritic" types from the ice morphology. For comparison, we provide photos of snowflakes that illustrate these same types. Although snowflakes form from vapor, not undercooled melts, the process of solidification is similar, and serve to show that our results are in close agreement with naturally occurring structures.

4.6. Banded Optimization

A good deal of the computation in the simulation is extraneous, because in many cases a large portion of the phase field grid is homogeneously ice or water. Eqn. 1 and 2 are only nonzero in regions where the phase field is heterogeneous, so any computation time spent in homogeneous regions is wasted.

Some of the optimization techniques that have been proposed for phase field methods include adaptive meshes for representing the phase and temperature fields [20] and Diffusion Monte Carlo (DMC) methods [19]. However, these techniques also deal with the accurate simulation of solidification at scales much smaller than the mesh resolution. Since we are only concerned with visual simulation, these smaller scales are not of interest to us.

The optimization that is of interest to us in the adaptive mesh and DMC methods is the localization of computation to the grid cells along the interface. This goal can be achieved using a simple method, similar to the "narrow band" optimization method used for level set methods [1]. Since all computation takes place using finite differencing, we know that the interface can move a maximum of one grid cell per iteration. If we restrict computation to grid cells that had a nonzero derivative on the previous iteration and their corresponding neighbors, then we will restrict computation of Eqn. 1 and 2 to only those grid cells that could potentially change. This simple and effective optimization offers the same computational localization as the adaptive mesh and DMC methods, while adding minimal implementation complexity.

Table 2 compares banded and unbanded performance. We used various resolutions of the stained glass window from Fig. 1 as our input. Although the performance is very input-sensitive, we believe Fig. 1 is a realistic input, since it initially covers about half the simulation domain.

Grid Size	Unbanded (Hz)	Banded (Hz)	Speedup
128 x 128	25	125	5.0x
256 x 256	8	20	2.5x
512 x 512	3.5	5	1.4x

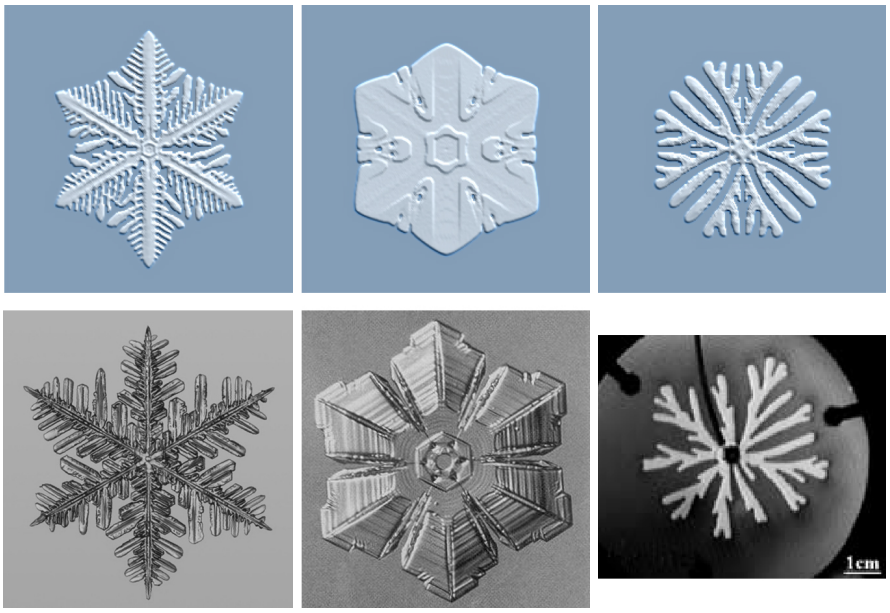
Table 2: Banded vs. Unbanded Performance

4.7. Hardware Implementation

Recently, the efficient solution of PDEs has become practical on programmable graphics hardware. Kobayashi's equations can be plugged directly into the general solution framework presented by Harris et al [11]. On a GeForceFX, we experienced as much as a factor of 9 speedup, making interactive simulation possible on non-trivial grid resolutions. Table 3 compares the two implementations. The timings are all for an unbanded implementation.

Grid Size	CPU (Hz)	GPU (Hz)	Speedup
64 x 64	250	624	2.50x
128 x 128	25	236	9.44x
256 x 256	8	67.47	8.43x
512 x 512	3.5	17.67	5.05x
1024 x 1024	1.08	3.77	3.49x

Table 3: CPU vs. GPU performance. CPU: 1.8 Ghz Pentium 4; GPU: GeForceFX 5800 Ultra



(a) (b) (c)

Figure 4: Top: Different simulated structures from the ice morphology. **Bottom:** Photographs for comparison. (a) *Dendritic* growth (b) *Sectored Plate* growth (c) *Isotropic* growth. Isotropic growth is not usually found in nature, because it is rare that no bias acts on growth. However, it can be produced in a laboratory using an electric field, as in this photo from Buka [3].

Banded optimization can also be implemented on hardware by terminating the fragment program as soon as the homogeneous phase case is detected. However, current GPUs do not yet support this functionality, so we are unable to obtain timing information that leverages this optimization.

5. User Control

One of our goals is to introduce a user parametrization into the simulation, so that a visual effects artist can suggest a general shape and the simulation can then grow an 'icy' version of the shape. The phase field method supports such a parameterization through the manipulation of its seed crystal and freezing temperature.

The seed crystal allows the user to guarantee that primary shape features are present in the final ice, and the freezing temperature allows the user to provide further simulation hints by rating the importance of secondary features. The settings for these parameters can be generated automatically using the methods suggested below, or interactively to give the user greater control over the final result. In order to illustrate how this works, we will grow ice in the shape of Fig. 5(a) as an example.

5.1. Seed Crystal Mapping

First, the user selects the most visually important features and maps them to the seed crystal. In this case, we decided that the edges of Fig. 5(a) were the most important visual feature. However, the user is free to select any arbitrary feature as the most important.

Fig. 5(b) was extracted using Canny edge detection [4]. We map these visually important features to the seed crystal to guarantee that these features are present in the final ice, preserving the general shape of the original image. However, if we then run the simulation on this seed crystal configuration, as shown in Fig. 5(c), the desired shape is quickly lost.

The seed crystal mapping only influences the initial condition of the simulation, so an additional parameter that influences the simulation at every timestep is also necessary in order for the goal shape to be preserved. The freezing temperature provides such a parameter.

5.2. Freezing Temperature Mapping

By varying the freezing temperature over the temperature field, we can model the presence of impurities in the undercooled liquid. Recall that salt causes ice to melt because it lowers the freezing temperature of the H_2O, and ice then transitions back to water if the surrounding environment is no longer cold enough to freeze it. Similarly, if salt were present in a undercooled liquid as it was freezing, the H_2O would be more reluctant to freeze in salty regions than in regions of pure water.

If we want to promote ice growth in a specific region of the phase field, we set the freezing temperature of that region as high as possible, to T_e. If we want to suppress all ice growth in a region, we set the temperature of that region to zero. To rate the importance of regions with respect to one other, we set their freezing temperatures between 0 and T_e.

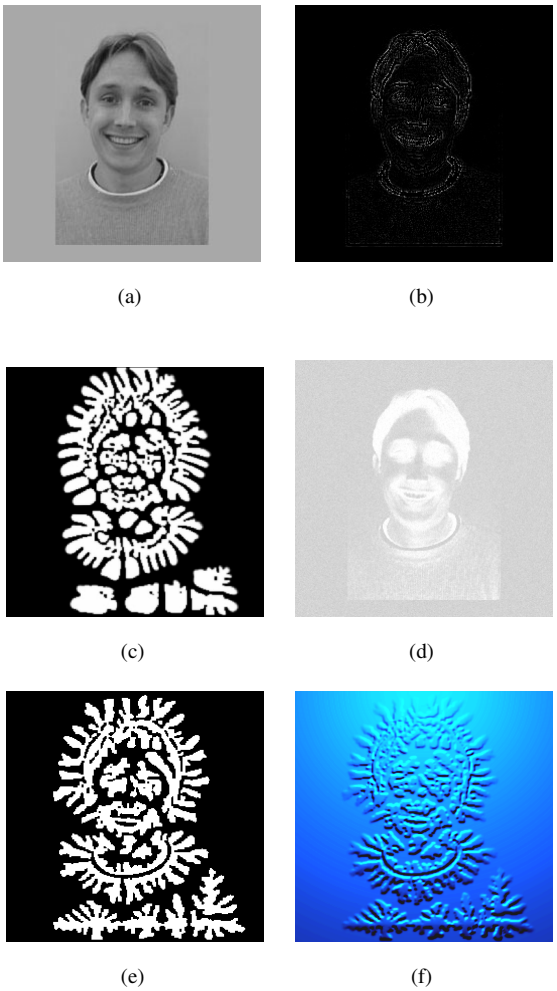


Figure 5: Left to right, top to bottom: (a) The source image; (b) the seed crystal texture; (c) simulation results after seed crystal mapping; (d) freezing temperature mapping: White regions are the original T_e value, while darker regions are lesser values; (e) the ice grown with a mapped seed crystal and freezing temperature; (f) the bump mapped ice.

For example, in Fig. 5(d), white regions represent Te, while greyer regions represent progressively lower freezing temperatures. The hair, eyes, and collar in Fig. 5(d) are whiter than their surrounding regions, so these regions freeze over first before the simulation starts branching out into the greyer regions.

We automatically generated the freezing temperature texture in Fig. 5(d) by first populating the texture uniformly with the default T_e values, and then subtracted a scaled version of the original image. This method rates dark regions higher than light regions and produced good results. However, this is only one rating method, and since the simulation can use any arbitrary texture, the user can impose any desired rating method.

6. Introducing Internal Structure

In this section, we introduce interesting, physically-inspired, internal structure to the results of the physically-based simulation. In the process, we will produce triangles from the results of the simulation that can be sent to a photon map renderer. This way we can capture one of the most striking features of ice, the caustics. Additionally, we will produce a subdivision surface representation that is capable of meeting the dense polygonal requirements of high-end visual effects work.

The phase field method provides the position of a growing ice border. However, there is also a good deal of interesting details that resides on the interior of ice as well. These details are apparent in the 'snowflake' images and simulations presented in Fig. 4. However, as we increase the scale of the simulation, these details are quickly lost, creating unnaturally flat ice.

6.1. Naïve bump mapping

In order to capture this internal detail, we first bump mapped the ice according to the $\frac{\partial p}{\partial t}$ of the ice. As water transitions to ice, it expands slightly, and this degree of expansion was approximated at each time step by increasing the height of the ice by the amount of phase transition. This is how the internal structures in Fig. 4 were produced.

However, this is a coarse approximation of the actual freezing process. The bumps in actual ice arise because of the expansion coefficient of water, which causes H_2O to increase slightly in volume as it freezes. This expansion coefficient arises due to forces at the water/air interface, not at the ice/water interface that $\frac{\partial p}{\partial t}$ is derived from. However, modeling the expansion coefficient is still an open problem in chemistry [21], and in our literature search we could not find a scientific model suitable for visual simulation and rendering. Consequently, we add a phenomenological step to introduce these additional features.

6.2. Adding Subdivision Creases

Once the simulation has run to completion, we can introduce sharp internal structures by inserting creases into the ice at visually expected locations. The introduction of creases into a surface is a well-studied technique in modeling, specifically using subdivision surfaces [6]. We introduce creases into the ice by stretching a subdivision surface over the ice and then repeatedly subdividing to introduce creases both at the border and along the medial axis.

The border is an obvious location to introduce detail, since it accentuates the border generated by the simulation. We introduce creases at the medial axis because we expect ice to have sharply faceted, crystalline features. Phenomenologically, the medial axis is a good candidate location for this crease because it is the location of visually interesting features in other natural growth phenomena, such as the veins in leaves. More formally, the medial axis is guaranteed to be distant from the border, so we are assured a good distribution of creases.

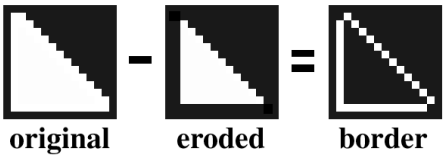
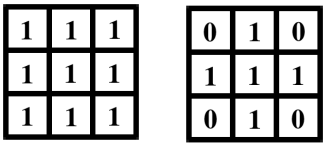


Figure 6: Border extraction operation



(a) (b)

Figure 7: (a) 3 x 3 structuring element for morphological erosion; (b) The sparser operator we use

6.3. Morphological Operators

We isolate both the border and medial axis through the use of morphological operators. This is a simple way to isolate both of these features, given that our final ice is stored as a nearly binary raster image. We can easily convert the image to a purely binary image by thresholding all $(p \geq 0.5)$ to 1 and $(p < 0.5)$ to 0. In addition, morphological operators guarantee connectivity properties that greatly simplify the construction of a subdivision control mesh.

Morphological operations can be viewed as binary convolution. In place of the multiplications and additions of normal convolution, we respectively perform logical ANDs and ORs. The convolution kernels in morphological operations are referred to as "structuring elements". See Jahne [13] for a more detailed description.

We use erosion, one of the simplest morphological operations, to isolate the border of the ice. If we run a single iteration of erosion on an image, then a single layer of white pixels around all white regions is deleted. If we then subtract this eroded image from our original image, we are left with the border pixels of all the white regions in the original image. This process is shown in Fig. 6.

The usual structuring element for erosion is given in Fig. 7(a). However, we use a sparser version, given in Fig. 7(b). As shown in Fig. 8, the use of the sparser structuring element does not give us the thick band of pixels that are present using the traditional element. Instead, we get a sparser set of pixels with simpler connectivity, which as we will see later, leads to a simpler subdivision control mesh.

We also use morphological operators to extract the medial axis of the ice. While there exist many ways to extract the medial axis, using morphological operators is very simple and guarantees the same connectivity properties as erosion, resulting in a simple subdivision control mesh. The use of morphological operators is slower than other medial axis algorithms, but the difference is negligible compared to the running time of the phase field simulation. Therefore, the extra overhead is insignificant in the overall computation time.

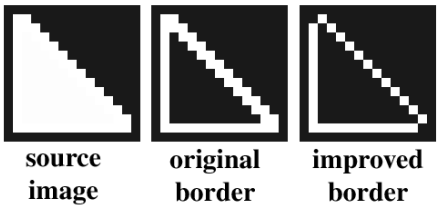


Figure 8: Results of modified erosion operator

In morphological terms, the isolation of the medial axis is known as the "skeletonization". The skeletonization operators in Fig. 9 are slightly more complex than the erosion operator. In addition to convolving by all eight structuring elements, the final value of the pixel is determined by ORing the results of all eight convolutions. The skeletonization operators also include "don't care" pixels. The image pixels that fall under the "don't care" pixels are ignored in all of the logical operations. In Fig. 9, the "don't care" pixels are denoted with empty pixels.

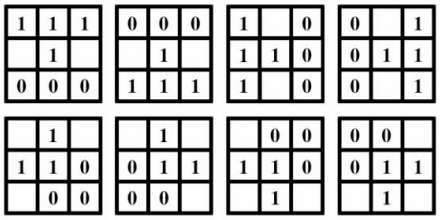
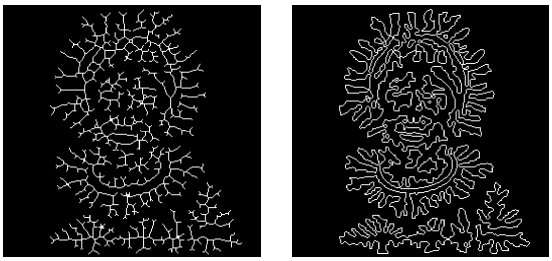


Figure 9: Skeletonization structuring elements

The structuring elements in Fig. 9 are each run repeatedly until no further changes take place. When this occurs, the pixels that remain are those along the medial axis. Note that the "thick" formations in Fig. 8 are also guaranteed not to occur in the skeletonized image. We obtained Fig. 10(a) and



(a) (b)

Figure 10: (a) Ice with skeletonization applied; (b) Ice with border operation applied

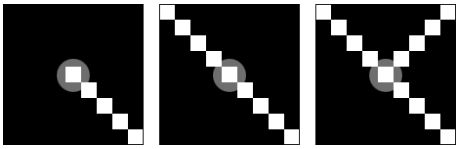
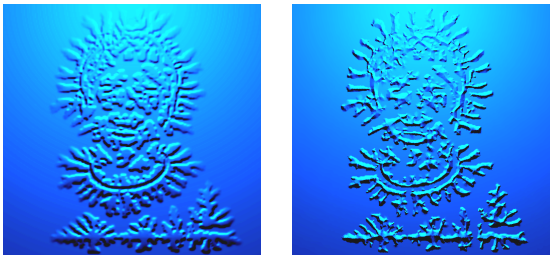


Figure 11: Crease pixel types (left to right): dart, crease, corner



(a) (b)

Figure 12: (a) The original bump mapped surface, (b) The sharpened ice surface after subdivision

(b) from our ice image, using the border and skeletonization morphological operators.

6.4. Control Mesh Segment Generation

To construct the control mesh for the subdivision surface, we first extract a set of line segments from the border and medial axis images. These line segments will be the crease edges in the subdivision surface, and their extraction is accomplished by performing a depth-first search of the white pixels in the images. According to Hoppe [12], there are three different vertex types at the endpoints of subdivision creases: "dart", "crease" and "corner" vertices. We can automatically tag our vertices to the correct type during the depth first search.

For the medial axis image, we can create a minimally connected mesh by exploiting the properties given by the morphological operators, as shown in Fig. 11. Any white pixel with only one white neighbor is a "dart" vertex, any white pixel with two white neighbors is a "crease" vertex, and any white pixel with more than two neighbors is a "corner" vertex. We run a similar algorithm on the border image, but since we are not guaranteed to have any dart pixels, we can start the traversal from any white pixel. Since there are not many darts or corners, we insert new vertices every so often, according to a "stride" length.

Note that if any of the "thick" structures from Fig. 8 had been present, then both dart and crease pixels would have two neighbors, and we would need a more complex search scheme. If we want to add more detail to the skeleton segments or avoid intersections with the border segments, we can add a "stride" to its tree traversal as well. In practice, setting the skeleton stride to the same as the border stride produced good results.

6.5. Triangulation Generation

Subdivision algorithms can only be run over tessellated surfaces, although the base primitive of the tessellation can vary. Several schemes can easily support creases, but we chose Loop subdivision because its base primitives are triangles, and there is a clearer path to generating triangles from the ice than generating other primitives.

A Delaunay triangulation algorithm takes a set of points and generates a set of triangles that contain the input points as vertices. In our case, we would also like to input a set of line segments, and generate a triangulation that contains both the points and lines. A specific variety of Delaunay triangulation, known as the "constrained Delaunay triangulation," accomplishes this task. In practice, the basic constrained Delaunay triangulation generated many "needle" triangles, so we used the constrained conforming Delaunay triangulation instead.

6.6. Height Field Generation

Once we have a two dimensional triangulation of the ice, we must assign height values to the vertices in the triangulation. The obvious choice is to sample values from the original bump map. However, since the original bump map is very smooth, the limit surface of a subdivision mesh based on its values can also be very smooth.

In order to guarantee the appearance of creases in the limit surface, we assign the height values according to a linear interpolation that approximates a faceted surface. We generate this approximation by first calculating the distance to the nearest border and medial axis pixels for all pixels. We then assign a height value to the pixel by linearly weighting the heights of the nearest border and medial pixels by their relative distance from the current pixel. The heights of the border and medial pixels are taken from the original bump map. This approximation is very much like the *contour connection* approach in [15] and is simply a linear interpolation between the medial axis and border contours.

Note that for more performance-driven applications, such as games, this height field can be used as a normal map in place of the more expensive subdivision surface representation.

6.7. Crease Generation

If the linear interpolation is used to set the height values of the triangulation, then the creases are present in the ice from the very beginning, and can be further refined through subdivision. As specified in [6], the creases can be made infinitely sharp or made arbitrarily smooth. If the mesh is already too dense for further subdivisions, then the vertices of the triangulation can be positioned directly to the limit surface, using the masks given in [12]. The results of our crease introduction step are shown in Fig. 12.

6.8. Rendering

Much of the interesting visual detail of ice is contained in the caustics generated by the refracting medium. To capture this

detail, we used photon mapping for rendering [14] the meshes generated by our detail reconstruction algorithm.

7. Implementation and Results

In this section, we give implementation details and present results generated on different scenes using our approach.

7.1. Implementation

All the pipeline stages were implemented in less than 5000 lines of C++ code, excluding the third party libraries cited below. Excluding the runtime library infrastructure, the hardware implementation took less than 100 lines of Cg code.

For our Constrained Delaunay Triangulations, we used Jonathan Shewchuk's *Triangle* package [23], a freely available Delaunay triangulation library that proved to be very well documented, easy to use, and highly optimized.

For rendering, we used POV-Ray 3.5, a freely available rendering application that supports a large shading language in addition to a nice photon map implementation.

7.2. Simulation Parameters

As mentioned earlier, the phase field simulation was run with the settings given in Table 1. The simulation ran successfully at the resolutions up to and including 2048 x 2048.

The time step was fixed to 0.0002 at all times. At larger steps, the numerical noise in the simulation quickly compounded. Other higher-order methods, such as Midpoint and Runga-Kutta Four integration, were attempted as well. However, they were unable to reliably increase the timestep size by a factor that would have justified their cost.

7.3. Results

We successfully simulated ice growth in several scenes. All simulations took place on a 512 x 512 grid with the exceptions of Fig. 14, which was 512 x 800. Our graphics hardware implementation runs at practically interactive rates, though its performance varies with the grid resolution. The first scene is a stained glass window, with ice growing inwards from the lead frame. Since the lead would cool faster than the glass, this seemed like a logical place to seed the ice. We ran the simulation for 600 iterations, taking a total of 34 seconds on a GeForceFX 5800 Ultra. The constant K was set to 1.2, and δ was set to 0.04. We also inserted a small amount of random noise into the freezing temperature map to promote non-uniform growth. Fig. 1 shows a detailed view on a portion of the stained glass with ice grown on it. See Fig. 15 and 16 for a sequence of snapshots from the simulation.

The second scene is a pond with ice growing on a lily pad, as shown in Fig. 13 (a). We ran this simulation for 800 iterations, taking a total of 45 seconds on the same GPU. The constant K was set to 1.2 and δ was set to 0.1. The ring example, shown in Fig. 13(b), was run on the same processor, for 2300 iterations, taking a total of 130 seconds. The constant K was set to 1.2 and δ was set to 0.1. A larger stained glass window with a more complex pattern is shown in Fig. 14. We ran this simulation for 500 iterations, taking a total of 50 seconds on the same processor. The constant K was set to 1.2 and δ was set to 0.1.

7.4. Discussions and Limitations

Validating the results of our simulation is very challenging, as the simulation is very sensitive to noise. In fact, it is this sensitivity that gives rise to such interesting structures. Very specialized equipment is necessary to run any meaningful experiments, which we unfortunately do not have access to. However, the physical validity of the phase field methods has been proven repeatedly by researchers in the computational physics and crystal growth communities who have access to such equipment.

Our technique and Diffusion Limited Aggregation [28, 2] both deal with the same basic problem of solidification. In practice we have found that our method can produce the same structures as DLA, and that these structures only represent a subset of those possible with our method. DLA also does not provide any clear way to introduce a user parameterization, and cannot achieve the simulation rates we achieve through graphics hardware. Given these factors, our method is a more practical technique for visual effects.

Finally, our reconstruction of the lost internal detail is only physically plausible, not physically based. Further study is necessary to validate and refine this process.

8. Summary and Future Work

We have presented a simulation technique from computational physics for the growth ice crystals and introduced optimizations to make the technique practical and interactive for computer graphics. We have also introduced a novel geometric sharpening operation to deal with the smoothing artifacts of the implicit simulation technique. Because ice growth has not been studied much in computer graphics in the past, there are many interesting future research directions.

The user parameterization we have presented is capable of preserving a desired shape, but the phase field model can support additional parameters for greater user control. The latent heat constant K, and the strength of anisotropy δ, both influence the growth speed and the final shape of the ice, and their spatial mapping could be used to achieve different effects. Mapping the θ_0 parameter could also be used to suggest shapes, such as ice growth in a spiral. The effect of external forces, such as gravity, wind, and fluid flow, are currently under investigation, and have the potential to produce more interesting results.

With respect to rendering, we assumed homogeneous ice when in fact ice can exhibit subsurface scattering, spatially variant densities, and contain pockets of air in the form of bubbles or cracks. These issues need to be addressed for the accurate rendering of ice.

The phase field method simulates several, but not all,

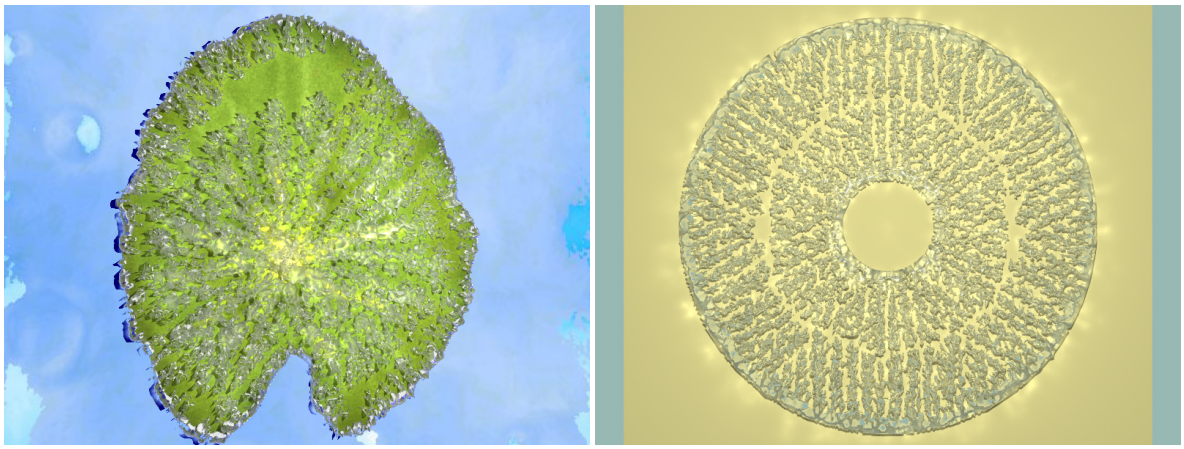
forms of ice crystal growth. These other types of ice crystal growth remain to be explored. Finally, the phase field method can be applied to fully 3D ice growth, capturing such phenomena as icicles. We plan to investigate other optimization techniques, such as parallel computation on a cluster of PCs, for efficient 3D simulation.

Acknowledgements

The authors would like to thank Konrad Kreszka for creating the church and lily pad models, Brian Utter from the Physics Department at Duke University for his informative discussions on solidification, and the anonymous reviewers for their suggestions. The stained glass patterns in Figs. 1, 15, 16, and 14 are from the *Art Nouveau Windows Stained Glass Coloring Book* [24] and appear courtesy of Dover Publications. This work was supported in part by Army Research Office, Intel Corporation, National Science Foundation, and Office of Naval Research.

References

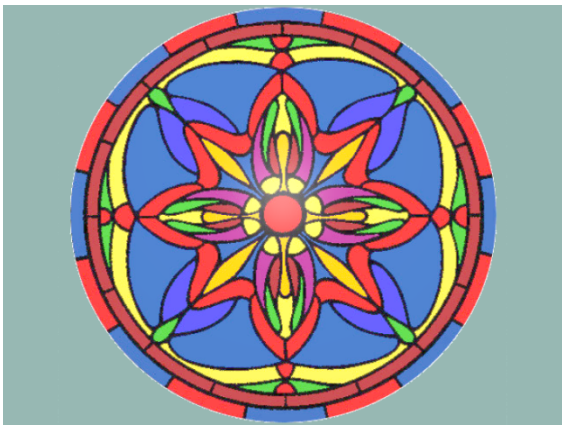
1. D. Adalsteinsson and J. Sethian. A fast level set method for propagating interfaces. *Journal of Computational Physics*, 118:pp. 269–277, 1995.
2. A. Barabási and H. Stanley. *Fractal Concepts in Surface Growth*. Cambridge University Press, 1995.
3. Á. Buka, T. Börzsonyi, N. Éber, and T. Tóth-Katona. Patterns in the bulk at the interface of liquid crystals. *Lecture Notes in Physics*, pages 298–318, 2001.
4. J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):pp. 679–698, 1986.
5. J. Demmel. *Applied Numerical Linear Algebra*. SIAM, 1997.
6. T. DeRose, M. Kass, and T. Troung. Subdivision surfaces in character animation. *Proc. of ACM SIGGRAPH*, 1998.
7. D. Enright, S. Marschner, and R. Fedkiw. Animation and rendering of complex water surfaces. *Proc. of SIGGRAPH*, pages pp. 736–744, 2002.
8. P. Fearing. Computer modeling of fallen snow. *Proc. of SIGGRAPH*, pages pp. 37–46, 2000.
9. N. Foster and R. Fedkiw. Practical animation of liquids. *Proc. of SIGGRAPH*, pages pp. 15–22, 2001.
10. F. Gibou, R. Fedkiw, R. Caflisch, and S. Osher. A level set approach for the numerical simulation of dendritic growth. *Journal of Scientific Computation*, (in press).
11. M. Harris, G. Coombe, G. Scheuermann, and A. Lastra. Physically-based visual simulation on graphics hardware. *SIGGRAPH/Eurographics Workshop on Graphics Hardware*, 2002.
12. H. Hoppe. *Surface reconstruction from unorganized points.* PhD thesis, University of Washington, 1994.
13. B. Jahne. *Digital Image Processing: Concepts, Algorithms, and Scientific Applications*. Springer Verlag, 1997.
14. H. Jensen. *Realistic Image Synthesis Using Photon Mapping*. AK Peters, 2001.
15. M. Jones and M. Chen. A new approach to the construction of surfaces from contour data. *Computer Graphics Forum*, 13(3):pp. 75–84, 1994.
16. R. Kobayashi. Modeling and numerical simulations of dendritic crystal growth. *Physica D*, 63:pp. 410–423, 1993.
17. B. Mandelbrot. *The Fractal Geometry of Nature*. W H Freeman, 1982.
18. S. J. Osher and R. P. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag, 2002.
19. M. Plapp and A. Karma. Multiscale finite-difference-diffusion-monte-carlo method for simulating dendritic solidification. *Journal of Computational Physics*, p. 165:592–619, 2000.
20. N. Provatas, N. Goldenfeld, and J. Dantzig. Adaptive mesh refinement computation of solidification microstructures using dynamic data structures. *Journal of Computational Physics*, 148:p. 265, 1999.
21. L. Rebelo, P. Debenedetti, and S. Sastry. Singularity-free interpretation of the thermodynamics of supercooled water ii. *Journal of Chemical Physics*, 109(2):pp. 626–633, 1998.
22. J. A. Sethian. *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge University Press, 1999.
23. J. R. Shewchuk. Triangle: Engineering a 2d quality mesh generator and Delaunay triangulator. In *First Workshop on Applied Computational Geometry*. Association for Computing Machinery, May 1996.
24. A.G. Smith. *Art Nouveau Windows Stained Glass Coloring Book*. Dover Publications, 1993.
25. G. Turk. Generating textures on arbitrary surfaces using reaction-diffusion. *Proc. of SIGGRAPH*, pages 289–298, 1991.
26. H. von Koch. Une mëthode géométrique élémentaire pour l'étude de certaines questions de la théorie des courbes planes. *Acta Mathematica*, 30:pp. 145–174, 1906.
27. A. Witkin and M. Kass. Reaction-diffusion textures. *Proc. of SIGGRAPH*, pages pp. 299–308, 1991.
28. T. Witten and L. Sander. Diffusion-limited aggregation, a kinetic critical phenomenon. *Physical Review Letters*, 47(19):pp. 1400–1403, 1981.
29. E. Yokoyama and T. Kuroda. Pattern formation in growth of snow crystals occurring in the surface kinetic process and the diffusion process. *Physical Review A*, page p. 41, 1990.



(a) (b)

Figure 13: ***Ice crystals grown on a lily pad and in a ring.*** (a) The entire simulation for the lily pad took 45 seconds. (b) The entire simulation for the ring took 130 seconds.

Figure 14: ***Ice crystals grown on a window panel.*** (Please view sideways.) Growth was started along the metal frame of the window. The entire simulation took 50 seconds.



(a)

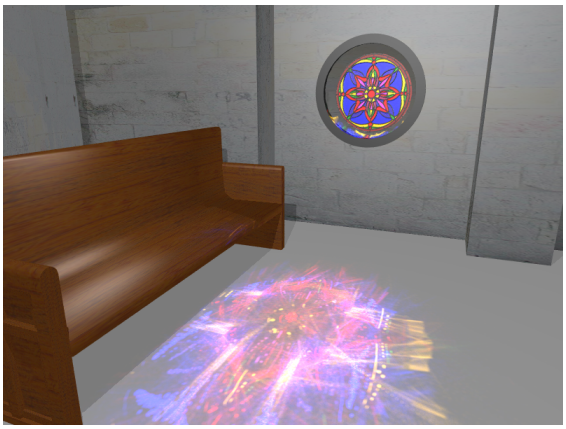
(b)

(c)

Figure 15: ***Ice growing on a stained glass window.*** Top to bottom: (a) The original stained glass; (b) After 340 iterations (c) After 540 iterations. Note how the ice crystals form starting from the lead frames. The entire simulation took 34 seconds.

(a)

(b)



(c)

Figure 16: ***Light refracting through a stained glass window.*** Top to bottom: (a) The original scene; (b) After 250 iterations (c) After 600 iterations. Note how the caustic changes as the refractive surface of th e ice becomes more complex.