Vision, Video and Graphics (2005)
E. Trucco, M. Chantler (Editors)

Modeling Falling and Accumulating Snow

T.B. Moeslund, C.B. Madsen, M. Aagaard, and D. Lerche

Computer Vision and Media Technology, Aalborg University, Denmark

Abstract

The use of computer graphics to produce special effects is currently being applied with great results in especially the entertainment and game industry. One area where computer graphics is not quite ready to replace all real effects is natural phenomena where a lack of general models exists. In this work we present a general model for falling and accumulating snow. The appearance and movement of falling snow are modeled in 3D based on the physics governing the real processes. The same goes for the accumulated snow where especially a correctly modeled wind field is important for producing realistically looking results. Intuitive weather parameters are used to control both models. The results show that both the appearance and movement of the snow, as well as the accumulated snow are very similar to real snow.

1. Introduction

The use of computer graphics (CG) in the entertainment industry is becoming an obvious choice in many situations. The main reason is the range of special effects that can be applied with great results. Just imagine movies such as "Jurassic Park", "Toy Story", and "Lord of the Rings" without the use of CG. Not to mention the entire game industry where CG is an inherent ingredient. However, other arguments for using CG also exist. For example, to reduce the production cost and risk of accidents by adding, e.g., artificial flames to a movie *after* the scene is shot.

One area where CG is not quite ready to replace all real effects is natural phenomena, e.g., the eyes of a human, fire, rain, and snow. The reason is first of all that the human observer knows how these phenomena look like in real life. Furthermore, some of the phenomena are not quite understood yet and therefore hard to model. Nevertheless, these and other natural phenomena have in some situations been modeled successfully using CG, but this success is mainly achieved using heuristic methods, for example visualizing a fire using a 2D billboarding technique. In general there is a lack of general comprehensive CG models of natural phenomena. In this work we present such a model of falling and accumulating snow.

1.1. Related Work

Previous work can be divided into nonphysically based models and physically based models. The nonphysical based models are methods that imitate the properties of a phenomenon without using the actual physics behind the phenomenon. As the actual physics are not used the methods are often simpler and less computationally heavy. Furthermore they are often ad hoc in the sense that they generalize poorly to a similar but different situation.

In [SW03] rain was simulated on videos. By analyzing videos of rain the authors concluded that it was not possible to track individual raindrops due to their high velocity. Using this knowledge the raindrops were modeled without any temporal coherency and by brightening the original image pixels at the position of the raindrops.

In [Sim90] a particle system with different specifications made it possible to simulate water, rain, snow, etc. An example is the simulation of a waterfall where the initial color of the particles was blue. As they hit a rock the particles bounced off and became white. Over time the particles would fade to the initial blue color. This gave the visual impression of water in a water fall by the use of simple color changes and particles bouncing off hard surfaces.

In [Fea00] a model for the accumulation of snow is presented. The primary focus was to model fallen snow where the layer of the fallen snow is thick. The calculation of the

fallen snow makes use of a particle system, where particles are launched from the ground and up. If a particle reaches the sky snow can be accumulated on the launch site.

When making a model based on physics a balance between the complexity and the visual result has to be found. It is not necessary to make a model that calculates the exact physics if no significant visual improvement is achieved. Therefore assumptions are often introduced in order to simplify the physical model.

In [FSJ01] a method based on Fluid Dynamics is presented which simulates smoke. A simplified version of the Navier-Stokes equations is applied. Introducing a rotational spin in the fluid keeps the smoke alive, and models turbulence in a visually correct manner. The method has the ability to interact with objects and thereby display a realistic flow.

In [NFJ02] a similar approach is followed to model fire. The method uses fuel present in a medium. When the temperature rises the fuel is converted into a hot gas which catches fire. The use of fuels gives the possibility to let the fire spread and ignite other flammable mediums.

In [FO02] a reduced version of the Navier-Stokes equations is used to model the accumulation of snow. The flow field within the scene is calculated while taking obstacles into account. Snow flakes are modeled as particles and dropped over the scene. These particles are influenced by the calculated wind field until they collide with a surface. The collision is registered and used to calculate the snow surface.

1.2. The Content of the Paper

In this paper we present a unified framework for modeling falling and accumulated snow. This includes a model of 3D snowflakes, a model of the forces governing the movements of a snowflake, and a model for how snow accumulates with respect to the forces and objects in the scene. All models are based on the underlying physics allowing a control of the different models via intuitive weather parameters, e.g., wind field, temperature, amount of snow per second.

The paper is structured as follows: in section 2 a model of a snowflake is described. In section 3 a movement model for a snowflake is derived, and in section 4 the primary force, the wind field, is described. In section 5 a model for accumulating snow is described. In section 6 results are presented and a conclusion is given.

2. Modeling a Snowflake

In this section we first outline the physical process resulting in real snow and then apply the main findings to model a snowflake.

The formation of a snowflake is a highly complicated process and not fully understood. However, it is known that a snowflake consists of ice crystals which can occur when five conditions are met. First of all the air in the atmosphere has to be saturated with water. This condition depends both on the temperature and on the vapor pressure. Secondly, the presence of condensation nuclei is necessary. These are small particles, e.g., dust or bacteria, on whose surfaces the water can condensate to produce small droplets. Thirdly, enough water has to be present in the cloud. When this is the case the droplets will keep growing due to further condensation or merging of droplets. When enough mass has accumulated the droplets will fall as precipitation. Fourthly, the temperature has to be below zero degrees Celsius in the cloud so that the water in the air will freeze onto the nuclei instead of condense into water droplets. The last condition concerns the nuclei that the water freezes onto (called freezing nuclei). They have to be of specific shapes so that the water molecules line up correctly to form the basic ice crystals. The demand on the shape of the nuclei decreases as the temperature decreases and at a temperature below -40 degrees nuclei are no longer necessary.

How ice crystals merge into actual snowflakes depends on the level of saturation and the temperature in the area between the cloud and the ground. The result is a virtually endless number of different snowflakes. They, however, follow some general shapes as seen in figure 1.

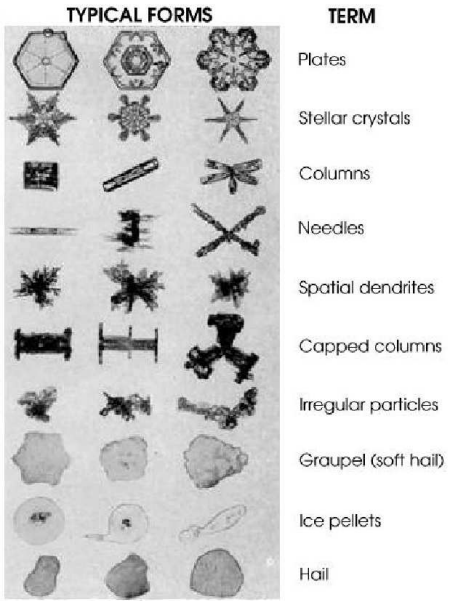

Figure 1: *Snow classification often used in hydrology [Lib04].*

2.1. Modeling the Size and Density

Looking at figure 1 and recalling own experiences with snow, the main characteristics to be modeled are the shape,

the size, and the density. All three characteristics are primarily controlled by the level of saturation in the air and the temperature. Modeling the saturation of the air is not necessary for rendering a scene, but it is necessary to model the temperature, as this can also affect other aspects of the scene. We therefore want a way of controlling the density and size of a snowflake using the temperature.

In the experimental work by [Jun00] the diameter of snowflakes has been measured as a function of temperature. From these results we derive the following relationship

$$D = \begin{cases} 0.015 \cdot |T|^{-0.35} & \text{for} \quad T \leq -0.061 \\ 0.04 & \text{for} \quad T > -0.061 \end{cases} \quad (1)$$

where D is the diameter in meters and T is the temperature in degree Celsius. Equation 1 represents the average diameter and from the experimental data [Jun00] we have seen uncertainties up to $\pm 50\%$. We therefore add a random number (within the limits set by the uncertainty) to the diameter from equation 1 whenever a snowflake is "born".

The density of snowflakes is inversely proportional to the diameter of the snowflakes: $\rho_{snowflake} = \frac{C}{D}$, and the proportionality constant, C, is $0.170 \frac{kg}{m^2}$ for dry snow and $0.724 \frac{kg}{m^2}$ for wet snow [RVCK98]. The distinction between dry and wet snow is defined to be minus one degree.

2.2. Modeling the Shape

Real snowflakes are primarily constructed by ice crystals colliding. Inspired by this fact we model a snowflake by combining triangular polygons in a random manner. We apply a number of concentric spheres to construct a snowflake. Each sphere is considered a layer and the same number of polygons is used for each layer. The number of layers directly gives the size of the snowflake and is controlled by the temperature as described above. By fixing the number of polygons per layer the density is also controlled by the temperature. An increase in the temperature results in an increase in the size which again results in a decrease in the density. We though have a different number of polygons for wet snow (40) and for dry snow (10). The number for wet snow is found empirically while the number for dry snow is found using the relationship between the proportional constants defined in the previous subsection.

When adding triangles to the different layers care must be taken to avoid free flowing triangles resulting in unrealistic structures. Therefore, when a new triangle is added to a layer one of its three corner coordinates must be connected with one of the triangles in the immediate inner layer, denoted the reference triangle.

First the reference triangle is randomly selected and then one of its corners is randomly selected. The spherical coordinates of this corner (θ_0, ϕ_0, r_0) defines the connected corner of the new triangle as $(\theta_1 = \theta_0, \phi_1 = \phi_0, r_1 < r_0)$, where r_1 is found randomly to be inside the reference triangle.

The remaining two corners of the new triangle, (θ_2, ϕ_2, r_2) and (θ_3, ϕ_3, r_3), are found randomly in the intervals: $\theta \in [\theta_0 - \varepsilon; \theta_0 + \varepsilon]$, $\phi \in [\phi_0 - \varepsilon; \phi_0 + \varepsilon]$, where ε is the allowed angular change (currently set to 80 degrees). The interval for the radius is within the immediate inner layer, the current layer, and the immediate outer layer.

In figure 2 a wet snowflake with a diameter of six cm (largest possible) is shown from two view angles and four different distances. The model of the snowflake has been parsed to the ray tracer POV-Ray, where it has been enhanced with transparent triangles and bump maps together with ambient and diffuse lighting. See [VID] for a 3D visualization.

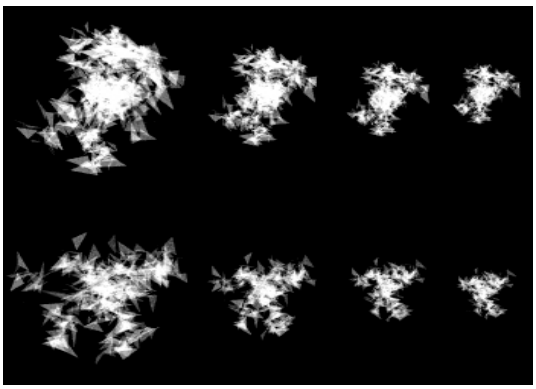

Figure 2: *A snowflake seen from two different view angles and from four different distances.*

To evaluate the effect of the density change due to the temperature figure 3 is consulted. The figure shows two wet snowflakes that are both produced at a temperature of -0.99 degrees, and two dry snowflakes both produced at -1.01 degrees. All four snowflakes have a diameter close to 1.5 cm and are displayed at two different distances. Observing the snowflakes it is clear that the wet ones are more compact as they have four times higher density. The dry snowflakes on the other hand have a much lighter appearance due to their small density.

3. The Movement of a Snowflake

The movement of a snowflake is caused by the four forces: $\mathbf{F}_{gravity}$ and $\mathbf{F}_{buoyant}$, which are both constant, and $\mathbf{F}_{lift}$ and $\mathbf{F}_{drag}$, which are perpendicular and change according to the wind direction.

$\mathbf{F}_{gravity}$ is the gravitational force and $\mathbf{F}_{buoyant}$ is the force that represents the up-drift by the surrounding air. The direction of the force, $\mathbf{F}_{buoyant}$, is always opposite $\mathbf{F}_{gravity}$. Furthermore, as $\mathbf{F}_{buoyant}$ in relatively small compared to $\mathbf{F}_{gravity}$

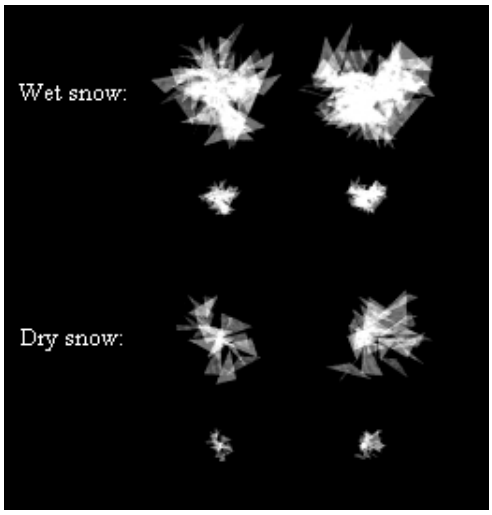

Figure 3: *Two wet and two dry snowflakes seen from two different distances.*

it can be ignored as it has no impact on the visual movement result.

The lift force, $\mathbf{F}_{lift}$, is the force that makes the snowflake move in circular and irregular patterns caused by the aerodynamic shape of the snowflake and the turbulence created behind the snowflake. At high wind speeds this force is insignificant, but in calm weather it cannot be ignored. For example, when observing a snowflake in calm weather one would observe that the snowflake follows the path of a helix towards the ground while rotating around its center of mass.

How these two rotations are carried out depends on the shape of the snowflake. As we model the shape randomly we will do the same for the angular rotation speeds. The rotational radius is also controlled by an initial random variable but the ratio between the current speed of the wind and the current speed of the snowflake is also taken into account. The consequence is that a large change in the wind will produce a large change in the rotational radius. See [AL04] for further details.

The drag force, $\mathbf{F}_{drag}$, represents the drag that the air will assert on the snowflake. This is the force that makes a snowflake follow the wind direction. The magnitude of the drag force can be expressed as [AL04]

$$F_{drag} = \frac{U_{fluid}^2 \cdot m_{snow} \cdot g}{U_{maximum}^2} \quad (2)$$

where m_{snow} is the mass of the snowflake, g is the gravitational acceleration, $U_{maximum}$ is the maximum vertical velocity taking wind resistance into consideration. For dry snow this is in the interval $[0.5m/s; 1.5m/s]$ and for wet snow it is $[1m/s; 2m/s]$ [Han99] [RVCK98]. When a snowflake is "born" we randomly sample from one of the two intervals according to the temperature. While m_{snow}, g, and $U_{maximum}$ are all constants for a particular snowflake, U_{fluid} is not. U_{fluid} is the speed of the air moving by the snowflake and the direction of U_{fluid} is also the direction of $\mathbf{F}_{drag}$.

$\mathbf{U}_{fluid}$ consists of two components; the wind velocity and the velocity of the air friction. The latter is the velocity of the snowflake, $\mathbf{U}_{snowflake}$, hence $\mathbf{U}_{fluid} = \mathbf{U}_{wind} - \mathbf{U}_{snowflake}$. $\mathbf{U}_{wind}$ is the velocity of the wind which obviously has a significant effect on the movement of the snowflake. In the next section this is further described.

4. The Wind Field

When snow is falling under the influence of a wind the snow will fly around obstacles in very distinct patterns that are caused by the wind field. These wind patterns are important to take into account when modeling falling snow in order to ensure that the snow falls correctly and the visual appearance resembles reality.

The wind field is a particular instant of Fluid Dynamics and can therefore be described by the Navier-Stokes equations. As the air in a wind field can be assumed to be incompressible, inviscid, and has a constant density of one, the Navier-Stokes equations can be simplified to the incompressible Euler equations [FSJ01]:

$$\nabla \cdot \mathbf{u} = 0 \quad (3)$$

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla)\mathbf{u} - \nabla p \quad (4)$$

where $\nabla\cdot$ is the divergence, $\mathbf{u}$ is a vector field of the velocity and p the pressure. Equation 3 states that the fluid conserves mass and equation 4 states that the fluid conserves momentum. The conservation of mass relates to the fact that the fluid is incompressible. Therefore the changes of the velocity in a well-defined area must equal zero as otherwise there would be a pressure change. The first term in equation 4, $-(\mathbf{u} \cdot \nabla)\mathbf{u}$, is the *convection* term which describes how the velocity of the fluid evolves over time. The second term, ∇p, is the acceleration of the fluid caused by the *pressure gradient*.

To solve the equations the solution is divided into two steps as it was done for the Navier-Stokes equations in [FSJ01] [Sta99]. First an intermediate velocity field $\mathbf{u}^*$ is calculated by using the Semi-Lagrangian scheme for the convection term. Afterwards the gradient of the pressure is calculated and used in a projection step where it is ensured that the mass is conserved according to equation 3. The two steps are repeated when the wind field needs updating†.

† Note that an off-line step is also present where a discretization of the scene into voxels is preformed [AL04].

4.1. Convection Step

This step will ensure that a small change in the air at a specific point will have an influence on the air in the rest of the domain. The propagation of the changes in the air is governed by the first term of equation 4, i.e., $-(\mathbf{u}\cdot\nabla)\mathbf{u}$.

Using a first order Taylor approximation $-(\mathbf{u}\cdot\nabla)\mathbf{u}$ can be calculated as [FSJ01]

$$-(\mathbf{u}\cdot\nabla)\mathbf{u} = \frac{\mathbf{u}^* - \mathbf{u}}{\Delta t} \tag{5}$$

where $\mathbf{u}^*$ is the intermediate velocity field and Δt is the time step between two updates. To calculate $\mathbf{u}^*$ the semi-Lagrangian method will be used [XXK02]. The method does not calculate the intermediate velocity directly. Instead it uses backwards-integration combined with interpolation in a fixed mesh, corresponding to the voxel faces. The calculation of $\mathbf{u}^*$ using backwards-integration is made with a time step of Δt.

Say we want to calculate the velocity at time $t+\Delta t$ for point P_a, see figure 4. We then trace back in time, to t, to see where a particle would had come from if it had the velocity $\mathbf{u}$ at time t. This gives us a so-called departure point, P_d, see figure 4. We then find the actual velocity of P_d at time t by trilinear interpolation and propagate this result forward in time. That is, the interpolated velocity is now the velocity at position P_a at time $t+\Delta t$.

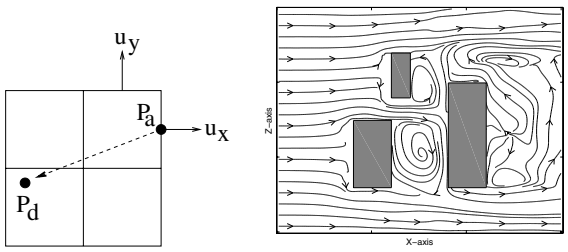

Figure 4: ***Left****: The principle of back-tracing in 2D.* ***Right****: A xz-slice of a wind field calculated for the scene in figure 7.*

In this work we apply a second order Runge-Kutta integration to perform the backwards-integration. For information on how the boundary conditions etc. are handled see [AL04].

4.2. Projection Step

The purpose of this step is to ensure conservation of mass of the fluid as stated in equation 3. This step is important as it ensures the creation of plausible wind fields with the correct swirly nature. In figure 5(left) a wind field is seen where only four voxels have velocities greater than zero (cones). Considering this wind field it is clear that it is not mass conserving as the wind suddenly starts in the middle of the wind field. This cannot occur for a mass conserving vector fluid as it means that air particles are moving from one voxel to another without being replaced and without pushing other air particles in the new voxel to the side.

In a more mathematical way the mass conserving property of the Euler equations states that the divergence should be zero. This means that the inflow into each voxel must equal the outflow, which is not the case in the figure. By applying the projection step (described in detail in the following) to the vector field in figure 5(left) the new wind field is given as seen in figure 5(right). In the figure it is observed how the wind is forced to loop back to conserve the mass within the wind field. The new wind field is much more turbulent/dynamic than the previous and it illustrates the importance of the projection step.

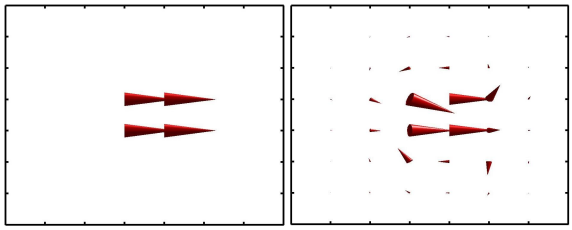

Figure 5: *A wind field before and after the projection step.*

The implementation of the projection is done using the Helmholtz-Hodge decomposition [Bet98] which states that any vector field can be decomposed into two parts as follows:

$$\mathbf{w} = \mathbf{v} + \nabla s \tag{6}$$

where $\mathbf{w}$ is a vector field without conservation of mass, $\mathbf{v}$ is a vector field with conservation of mass ($\nabla\cdot\mathbf{v}=0$) and ∇s is the gradient of a scalar field. This property can be used for the projection of $\mathbf{u}^*$ into a mass conserving vector field by subtracting the gradient of a scalar field. In our case where the vector field is a velocity for a fluid the scalar field is equal to the pressure field for the wind. Therefore the projection is given by:

$$\mathbf{u} = \mathbf{u}^* - \nabla p \tag{7}$$

Following along this line of reasoning the pressure at each voxel can be expressed as a Poisson equation where the Neuman boundary condition is used [AL04]. Solving for all voxels results in a linear system where the coefficient matrix is spares. We apply the Conjugate Gradient Method [FF01] to solve this linear system. In figure 4 an example of a wind field is illustrated. See [VID] for videos.

5. Modeling Accumulated Snow

Using the model for falling snow described above we could simply drop snowflakes from the sky and count the number of times a particular position in the scene is hit by a snowflake. Smoothing the result and we have a scene with accumulated snow. However, this approach will require a tremendous number of snowflakes, i.e., very long processing time. Instead we want to be able to speed up this process so that the designer can define the amount (height) of snow and produce a result "immediately". We use a six step algorithm for this purpose.

The first step divides the scene into larger entities, denoted edge groups, having the same orientation. The next four steps create the accumulated snow layers in the scene in an iterative manner. By doing so it will be possible to update the wind field for each iteration and thereby include the effects of the changing wind field into the model for the accumulated snow [FO02]. Furthermore, it allows for intermediate results of the accumulating snow, i.e., the process can be canceled halfway through and still produce a realistic result. The last step renders the surface of the accumulated snow. In the following each step is further described.

5.1. Create Edge Groups

The first step in the accumulation scheme is the creation of the edge groups. This step functions as an initialization and is basically a question of determining possible locations where snow can accumulate. Each edge group consists of a series of triangles that all belong to the same object and where all triangles know their neighboring triangles or whether or not they lie along an edge. All the triangles within an edge group is bounded by a surrounding edge, hence the name edge group. An edge group is an isolated part of the scene where it may be possible for snow to accumulate.

5.2. Emit Snow Particles

In order to make the accumulation it is necessary to have a representation of the snowflakes and to know where they will fall on the mesh consisting of triangles. We emit snow from the "sky" and calculate with which triangle in the edge groups there is a collision. To ensure realism the emitting and movement of the snowflakes use the schemes described in the previous sections. As the snowflakes are not visualized during fall we do not model their appearance and therefore refer to them as snow particles. To speed up the process each particle is assigned a volume of $10^{-4}m^3$ per particle. For further details on emitting particles and collision detection see [AL04].

5.3. Refine Edge Groups

A very detailed snow surface is important when representing the snow around obstacles as the snow height will vary a great deal, however, it is not always necessary to have a detailed representation of the snow surface on, e.g., a large plane surface where no obstacles exist. We therefore divide the triangles into a finer grid of triangles according to the number of particles hitting the triangles. Concretely we compare the center of each triangle with the center for the snow particles hitting this triangle and divide accordingly. In figure 6 the triangles are shown before and after this refinement process. See [Fea00] [AL04] for details.

5.4. Resolve Stability

The next step is used for resolving the stability. There are two purposes of this step. The first is that it functions as a smoothing mechanism that makes sure that too abrupt transitions between two levels of accumulated snow do not occur. The second purpose of the stability step is the most important as it determines if the snow cover is stable enough. This is important as it for example determines whether or not a snow cover on a rooftop is too high to remain stable or falls down. Another example is whether or not a snowdrift stays stable when leaning against an object.

We apply the algorithm from [Fea00] where the idea is to redistribute snow from one triangle to its neighbors if the height difference is too steep. This is done in an iterative manner based on a list of all triangles sorted in descending order. See [Fea00] [AL04] for further details.

The height difference between neighboring triangles is expressed as an angle and for snow to be distributed this angle has to be above the angle of repose (AOR). The AOR is dependent on various phenomena such as the roughness of the surface, the type of snow, and the temperature [GM81] [MS93]. However, no equation for the AOR can be found and therefore one is derived based on the experiments made in [Fea00]. Using these results we end up with the following relationship between the AOR and the temperature. It should be stressed that this equation is merely a crude approximation.

$$AOR = \begin{cases} 30 \cdot |T+6|^{-0.25} + 40 & \text{for} \quad T \leq -8.5 \\ \frac{26.1418}{8.5} \cdot T + 90 & \text{for} \quad T > -8.5 \end{cases} \tag{8}$$

5.5. Smoothing and Rendering the Surface

The final step in the iteration is to smooth the snow surface. This is done by averaging the height of each triangle using its adjacent neighbors and then connecting these averaged centers. After the iterations are done a smoothed surface with the correct height according to the specified amount of falling snow, the wind field and the objects in the scene is obtained. The final result is rendered using POV-Ray.

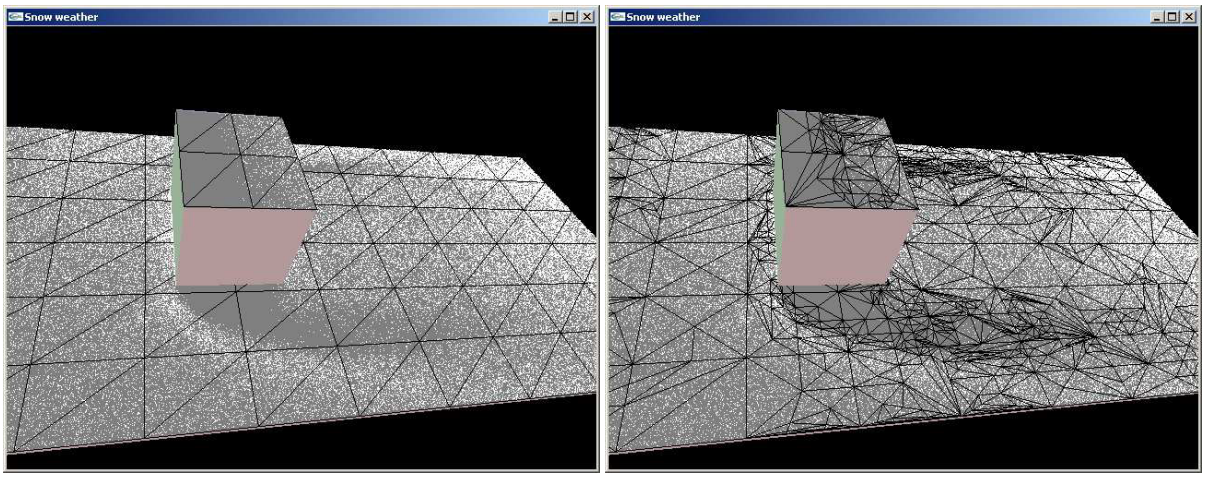

Figure 6: *The grid before and after the refinement step.*

6. Results and Discussion

In this work three main topics have been covered: modeling a snowflake, modeling the movement of a snowflake, and modeling the accumulation of snow. In this section we present some results regarding these three topics and discuss them.

The appearance of the snowflake, which is illustrated in section 2, is believed to provide a realistic appearance. In general the snowflakes have a nice fluffy structure resembling real snowflakes also compared to the CG snow currently used in movies. It is only when a snowflake comes extremely close to the "camera" that it might seem unnatural. This is a general problem of artificially created objects and therefore often avoided altogether in movies. It is still to be seen whether a very good lighting model for our snowflakes will solve this problem. The effect of the temperature is clearly observable as seen in figure 3, see [VID] for videos. Regarding processing time we can update the position of around 100.000 snowflakes per second and it scales linearly[‡].

In many situations the movements of the snow as a whole will be more important than the appearance in order to achieve a believable result. For example, in one of the videos [VID] the snowflakes are modeled using simple shapes with different sizes, but nevertheless the snow seems realistic at first glance. Note that this particular video has been generated in real-time due to its simplicity!

Our movement model is based on the actual forces governing the physical processes and is therefore very realistic. In figure 4 a top view of a wind field is shown and it can be seen that the expected turbulence appears nicely compared to the objects in the scene, see figure 7. This is also documented in the videos [VID]. Besides the overall wind field each snowflake also rotates with respect to its center of mass providing a chaotic swirly nature very similar to real snowflakes.

The velocity of the wind has a large effect and can easily be controlled by an operator. In fact, we have a simple (in terms of the shape of the snowflakes and the scene) online version of our system where a wind gust can be entered via the keyboard, see [VID] for videos.

The discretization of the scene can have a significant influence on the wind field. However, this is not as profound as we initially thought. Only when you know what to look for a difference can be observed, see [VID] for videos. It is noticed that the smaller the discretization the more complex the path of the snowflakes. The system can approximately update a wind field containing 27.000 voxels per second and it scales linearly.

Using the wind field shown in figure 4 the accumulated snow in figure 7 is achieved. The result is in general similar to what should be expected in a real situation. However, still some parameter tuning is required in order to avoid too high snowdrifts along the walls, see [VID] for video. The processing power is in minutes depending on a number of issues, such as how many triangles are present after the refinement process, and how many iterations are required to achieve stability [AL04]. In figure 7 an example of our unified model can be seen, i.e., both accumulated and falling snow, see [VID] for video.

6.1. Conclusion

The usage of the three elements described in this paper (snowflake, movement of snow, and accumulation of snow)

‡ Using non-optimized code on a 1.8GHz PC with 256MB Ram.

Figure 7: ***Top****: Accumulated snow using the wind field in figure 4.* ***Bottom:*** *Combining falling and accumulated snow.*

is the following. The model of a snowflake can be used to add artificial snow to a scene. Since we are applying a 3D model as opposed to a 2D billboarding technique, the appearance of the snow is independent of the camera's viewpoint. The snow appears real both regarding to its appearance (fluffiness and density) and its movement patterns (follow wind, gravity, and self-rotating). The model is controlled using the main weather parameters: temperature and wind field, hence the model is easily adopted to a new scene. Furthermore, since the wind field is general the snow movement also seems natural with respect to the objects in the scene.

The accumulation of snow can be used to provide layers of snow reflecting the wind field and the objects in the scene. Again a few weather parameters can be used to control the entire model. The model for the accumulation of snow and the model for falling snow can obviously be combined.

In conclusion is can be stated that both the appearance and movement of the snow, and the accumulated snow behave very much like real snow and can be used in any static scene since both the wind field and the other control parameters are general.

References

[AL04] AAGAARD M., LERCHE D.: *Realistic Modelling of Falling and Accumulating Snow*. Master's thesis, Aalborg University, Denmark, 2004.

[Bet98] BETOUNES D.: *Partial Differential Equations for Computational Science*. Springer-Verlag, 1998.

[Fea00] FEARING P.: *The Computer Modelling of Fallen Snow*. PhD thesis, University of British Columbia, 2000.

[FF01] FOSTER N., FEDKIW R.: Practical Animation of Liquids. In *Proceedings of SIGGRAPH* (2001).

[FO02] FELDMAN B., O'BRIEN J.: Modeling the Accumulation of Wind-Driven Snow. In *Conference Abstracts and Applications SIGGRAPH* (2002).

[FSJ01] FEDKIW R., STAM J., JENSEN H.: Visual Simulation of Smoke. In *Proceedings of SIGGRAPH* (2001).

[GM81] GRAY D., MALE D. (Eds.): *Handbook of Snow*. Pergamon Press, 1981.

[Han99] HANESCH M.: *Fall Velocity and Shape of Snowflakes*. PhD thesis, Ludwig-Maximilians University, Munich, Germany, 1999.

[Jun00] JUNKER N.: Winter Weather Forecasting. *www.hpc.ncep.noaa.gov/research/snow2a/*, 2000.

[Lib04] LIBBRECTH K.: A Field Guide to Falling Snow. http://www.its.caltech.edu/˜atomic/snowcrystals/class/snowcrystals.pdf, 2004.

[MS93] MCCLUNG D., SCHAERER P.: *The Avalanche Handbook*. The Mountaineers, 1993.

[NFJ02] NGUYEN D., FEDKIW R., JENSEN H.: Physically Based Modeling and Animation of Fire. In *Proceedings of SIGGRAPH* (2002).

[RVCK98] RASMUSSEN R., VIVEKANANDAN J., COLE J., KARPLUS E.: *Theoretical Considerations in the Estimation of Snowfall Rate Using Visibility*. Tech. rep., The National Center for Atmospheric Research, Canada, November 1998.

[Sim90] SIMS K.: Particle Animation and Rendering Using Data Parallel Computation. *Computer Graphics 24*, 4 (August 1990), 405–413.

[Sta99] STAM J.: Stable Fluids. In *Proceedings of SIGGRAPH* (1999).

[SW03] STARIK S., WERMAN M.: Simulation of Rain in Videos. In *The 3rd international workshop on texture analysis and synthesis* (October 2003), pp. 95–100.

[VID] Videos: http://www.cvmt.dk/Snow/.

[XXK02] XU J., XIU D., KARNIADAKIS G.: A Semi-lagrangian Method for Turbulence Simulations Using Mixed Spectral Discretizations. *Journal of Scientific Computing 17* (December 2002), 585–597.