Eurographics Workshop on Natural Phenomena (2009)
E. Galin and J. Schneider (Editors)

A Geometric Algorithm for Snow Distribution in Virtual Scenes

N. v. Festenberg[1], S. Gumhold[1]

[1] Lehrstuhl für Computergraphik und Visualisierung, TU Dresden, Germany

Abstract

Freshly fallen snow is a popular natural phenomenon able to evoke great beauty in all kinds of scenes. However, there still does not exist an all-purpose algorithm for automated snow distribution in virtual worlds. Previous works modelled snow either relying on costly particle simulations or oversimplified surface displacements. In this paper we develop a novel geometric snow model. In a first step, we propose a statistical snow deposition model inspired by real world observations. This statistical model is used to derive a geometric snow shape formulation. The geometric scheme is implemented with an enhanced height span map. Scene geometry is expressed with two parameter fields that enable us to efficiently compute snow cover geometry.
A selection of snow covered scenes demonstrates the realism that can be achieved with this new method.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.6]: Simulation and Modeling—, Computer Graphics [I.3.7]: Three-Dimensional Graphics and Realism—

1. Introduction

It is an ever ongoing quest to increase realism in virtual worlds. Increased realism requires transferring more and more effects and phenomena from the real world into computable schemes. A wide range of phenomena has already been reproduced in photorealistic quality (e.g. smoke as in [SB08], or water [IGLF06] or plants [DL05]).

However, there are many phenomena where automated reproduction could not be achieved yet. Often, manual modelling is not feasible or too expensive. One of these phenomena - ubiquitous at least on the northern hemisphere in winter - is snow. Snow is a powerful means of decoration in virtual environments. The addition of a layer of frozen water can transform scenes drastically evoking an enchanted fairy tale ambiance. Snow distribution is also a subtle geometrical problem. The problem is principally related to casting light on a scene. The resulting snow distribution borders are similar to soft shadows. Though, second order effects in light as reflection or refraction do not directly translate to snow accumulation. Instead, in snow one finds complex superpositions of lateral redistribution. When snow falls on a conifer for example it will be deflected several times by branches and needles of different height levels before it settles. The range of phenomena in snow accumulation is vast. Even a square inch of snow is formed out of thousands of snow flakes all different in shape and each far from being isometric or smooth [Nak54]. This complexity is aggravated by the average winter temperatures in mid-latitudes close to the melting point of snow. Crust formation can occur due to subsequent melting and freezing as well as volume reduction as consequence of local recrystallization and air removal.

This justifies the statement that snow is one of the most complex materials in the world [GM81]. Until today, even elaborate models with little limitations in computation time fail to reliably reproduce snow behaviour, e.g. in avalanches [Sto05]. Nevertheless, it is possible to formulate useful approximate models designed to meet the requirements of a certain context.

In this paper we propose a novel geometric algorithm to compute realistic snow covers for virtual scenes. We focus on the snow distribution pattern in the scene. We do not animate snow fall or develop methods for a scene's elastic feedback to a snow cover nor did we consider particular optical features of snow. The core innovation is our model's foundation on real snow observations. We condensed these observations into a statistical snow deposition model allowing for the computation of an approximate snow mass probability distribution. This distribution is implemented as geometric scheme. We explicitly compute the three dimensional shape of the snow cover delivering the fundamentals for later manipulation in animations as e.g. foot prints. A wide range of scenes and lateral transport scenarios is covered. By aiming at an deterministic model we widely avoid stochastic computations keeping the algorithm efficient.

Section 2 gives a brief overview on previous work on modelling snow and related phenomena. In Section 3 we describe the assumptions of our snow accumulation model and explain its mathematical base. In Section 4 we show how to implement our algorithm and in Section 5 we present our results. Discussion and comparison of our results as well as possible improvements of our algorithm are given in Section 6 and 7.

2. Previous Work on Snow Modelling

There exists some literature on snow models in physics and environmental engineering. Please refer to [Sto05, Mel74] for further details.

In Computer Graphics modelling snow has received relatively little attention. The first model for fallen snow was proposed by Nishita et al. in 1997 [NIDN97] based on metaballs and user specified snow distribution. Subsequent model proposals can be divided into two groups: models based on particle simulations on the one hand, and surface displacement models on the other hand. The most influential model from the first group was published by Paul Fearing in 2000 [Fea00]. The model aims at computing the shape of fallen snow with particles shot upwards and recursive snow surface stability tests. Moeslund et al. [MMAL05] proposed a revised combination of the Fearing model with animated snow fall, but still realism was not reached. Several papers were published on dynamic features of snow, such as wind driven transport [FO02, WWXP06] or snowflake animation [LZK*04]. There were also attempts to parallelize snow simulations [SEN08].

Surface displacement models from the second group are generally faster in snow cover computation at the expense of realism. The proposals in [PTS99] and [OS04, Dud05] present convincing results for large scale scenes based on surface texturing techniques. Another large scale snow model borrowing methods from illumination was proposed by Foldes et al. [FB07].

To our knowledge, most realistic results were presented by Muraoka et al. [MC00] with a model outside the two model groups. Their model is derived from the microscopic physical properties of snow and water. Both snow fall, snow cover and snow melt can be expressed. Unfortunately, little implementation details were published.

An approach close to our implementation was introduced by Tokoi [Tok06]. Similar to [OS04] the author uses the Z-buffer to determine snow locations, but he finds smooth borders with the help of several slightly perturbed depth textures as in soft shadow computation. A simple particle based stability test is applied. Still - partly due to a manual preprocessing step - performance is high.

Our model is derived from a physical snow deposition model. Different from preceding works, we use photographs of snow covered scenes as primary source for model development, i.e. we exploit visual scale shape measurements of snow to formulate a statistical deposition model. We fit the geometric essence of this model into an efficient data structure called height span map. Sumner et al. [SOH99] originally proposed this structure for interactive animation of granular material and Onoue et al. [ON05] presented an improved version towards real-time applicability. Haglund et al. [HAH02] connected the height span method with snow to model real-time snow accumulation, but without great detail in the distribution characteristics.

We will show here that also realistic snow distribution schemes can be adapted beneficially to a height span map. By combining real world observations with an efficient model architecture we present a snow distribution algorithm both more realistic and more extensible than previous models.

3. Geometrically distributed Snow

A one-crystal snow flake has a volume of ca. $10^{-9}\mathrm{m}^3$ [Nak54]. Snow crystals contribute about 10 percent of the volume of freshly fallen snow, the rest is mainly air [Sto05]. This means that around ten million snow flakes are needed to cover one square meter ground with ten centimeters of snow. The snow flakes themselves appear as widely varying dendritic stars complicating simulation.

Fortunately, the complex microscopic structure does not disturb a stable appearance of snow on visual scales. When we derive our geometric model, we rely on this visual scale as observed in common photographs. For simplicity, we assume that the amount of snow deposited per sky area is constant all over the simulated scene. Modelling snow in a scene consisting of an infinite plane is trivial then: snow height will be constant everywhere. Characteristic snow shapes will occur near or below edges, because there snow is forced to redistribute. In Figures 1, 2, 5 and 8 photographs of snow covers in real world scenes are shown. You can see that near the edges of a snow body the snow cover defines a smooth curve from almost vertical to horizontal angles. Obviously, no specific angle of repose as stated elsewhere (e.g. [Fea00]) can be observed. Note that thicker snow covers as in 1b coincide with steeper edge curves.

When looking at more of these edge curves in the real world in different weather conditions one can discover a great variety of possible shapes, sometimes even with overhangs. Despite the variety the principal shape remains the

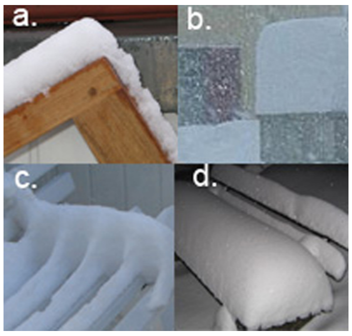

Figure 1: *Snow photographs for different snow conditions.*

same: steep angles close to an edge smoothly evolving to horizontal angles far from the edge.

We will now propose an explanation for this profile curve. When a snow flake hits a surface it might stay there, bounce off or slide along the surface depending on different factors. On the one hand these factors will be physical properties such as snow wetness, surface roughness and both snow and surface temperatures. On the other hand geometric constraints as snow flake velocity and surface orientation will play an important role. For simplicity (and feasibility) we confine ourselves to the consideration of geometric constraints and otherwise assume as little as possible. Let $\vec{v}$ de-

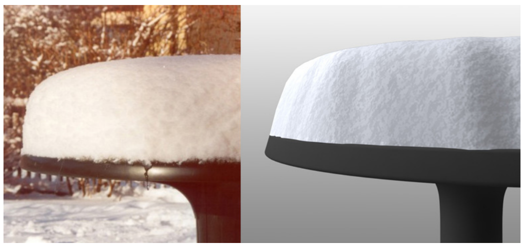

Figure 2: *Left: Snow covered table. Right: model with snow reproduced with our method. Grid resolution 60×60, $h_{max} = 0.25$,c=8,$\alpha_m = 20°$, one smoothing pass, computation time ca. 70 ms. Discretization artefacts could easily be removed but were left for clarity.*

fine the velocity of a snow flake in a scene with an empty, horizontal surface S with curved boundary ∂S. The velocity is split into the vertical component $\vec{v}_\perp$ and the horizontal component $\vec{v}_{||}$. We assume that $\vec{v}_\perp$ is constant for all snow flakes while $\vec{v}_{||}$ varies according to a normal distribution with zero mean (i.e. we assume the snow flakes to tumble through air without directional wind). When a snow flake hits S at location $\vec{x} \in S$ we assume $\vec{v}_\perp$ to be annihilated. The snow flake will slide along the surface before it settles, and the length of its path then depends on $\vec{v}_{||}$ only; and we assume it will be proportional. So if $\vec{x}$ is close to ∂S the snow flake will probably leave S again. We can quantify this snow distribution probability delivering a statistical approximation of the future snow height. The amount of snow flakes a location $\vec{x}$ will receive depends on the area around $\vec{x}$ inside ∂S. Formally, it is the probability weighted sum of all flakes that finish their slide exactly at $\vec{x}$. Thus, we simply need to integrate a bivariate normal distribution centered at $\vec{x} = (x, y)$ over S to obtain a measure of the snow height $h(\vec{x})$ as illustrated in Fig. 3. As we did not fix the scaling yet we can assume standard deviations equal to one without loss of generality, yielding

$$h^{\text{exact}}(\vec{x}) = h_{\max} \cdot \frac{1}{2\pi} \iint_S e^{\frac{-(x'-x)^2-(y'-y)^2}{2}} dx'dy' \qquad (1)$$

which was not proposed before to our knowledge. For a straight edge bounding a very large plane the snow height h^{plane} depends only on the distance $x = 0..\infty$ to the edge and we can solve (1) exactly

$$h^{\text{plane}}(x) = h_{\max}(0.5 + \text{erf}(x)), \qquad (2)$$

with the error function $\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-x'^2} dx'$ and $h_{\max}$ the maximal snow height far from edges. This curve is depicted in Fig. 4. It is also valid for surfaces where the minimal diameter is large compared to the horizontal snow velocity, i.e. the standard deviations in (1). This of course is an idealization, but it is a useful foundation to efficiently approximate snow height in visual simulations. Note that moderately curved or tilted surfaces do not change the statement (1) qualitatively.

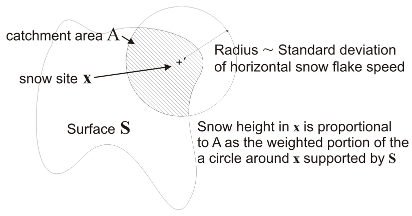
catchment area A
snow site x
Radius ~ Standard deviation of horizontal snow flake speed
Surface S
Snow height in x is proportional to A as the weighted portion of the a circle around x supported by S

Figure 3: *Sketch of the statistical snow deposition model: the snow height depends on the portion of the catchment area around a snow location that is supported by the underlying surface. The radius is proportional the standard deviation of the snow flakes horizontal speed.*

In general, (1) will be cumbrous to integrate exactly. We therefore use (2) as approximation by defining the snow height h to depend on the scalar distance to the nearest edge for all surface shapes. We found the polynomial

$$h(x) = h_{\max}\left(1 - (cx - 1)^4\right) \qquad (3)$$

to be a good computable approximation of (2), though higher order terms might be added, too. Here, x is proportional to the distance from the snow edge valid for $0 \leq cx \leq 1$. The factor c is a real number between zero and infinity used to model different steepnesses of the edge curve, i.e. different snow qualities or different standard deviations of the horizontal snow flake velocity. For example $c \gg 0$ would represent an almost perpendicular edge observed in snow fallen in large snow flakes. Figure 4 shows a selection of different edge curves according to (3).

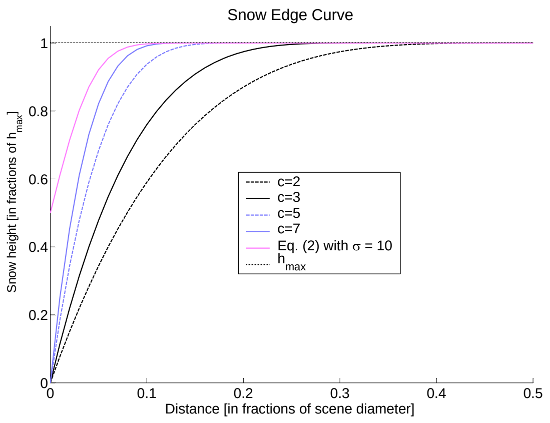

Figure 4: *Edge curves as in eq. 3. The factor c is given in fractions of the scene diameter, i.e. c =5 means that maximum snow height is reached within 20% of the scene length.*

The ideal edge curve is not always realized. First, S might be curved with respect to the vertical direction. Fig. 1b indicates that steeper slopes as e.g. old snow surfaces close to edges tend to decrease snow deposition probability creating smaller snow heights or edge profiles steeper than (2). Second, the maximal snow height is never reached if snow patches are very narrow as in Fig. 1c. We focus on two requirements that must be met for an undisturbed edge curve: A) the sky above that edge is not occluded by objects above the edge, i.e. the sky can be reached on straight lines and B) the patch defined by a closed edge is large enough to carry the full amount of snow fallen from the sky. Obviously, this is true only for simple scenes. Moreover, we need to redistribute the snow missing in the edge regions to guarantee snow mass conservation. We will show now how to conserve the snow mass when requirements A and B are not fulfilled. Again, real world observations and (1) deliver essential insights. We first need to define what a snow patch is. Generally speaking - though not true in exotic cases neglected here - snow can only settle on surfaces with a normal direction $\vec{n}$ smaller than 90° with respect to the direction of snow fall $\vec{f}$. This defines a set of candidate locations for snow deposition. However, gentler slopes are required to allow fallen snow to form contingent snow patches. Let S_i be one of these snow patches, i.e. a subset of the candidate locations. It can be defined as union of all connected infinitesimal surface segments dS with local angles $\alpha := \angle(\vec{n}(dS), \vec{f})$ less then some limit $\alpha_m < 90°$

$$S_i = \{\text{all connected } dS \text{ with } \alpha < \alpha_m\}. \tag{4}$$

For simplicity, we choose to neglect potential snow depth reduction in steep slopes, which is reasonable when we treat h as horizontal snow depth and not as snow thickness. We can further assign a radius $r_{\max,i}$ to each patch denoting the patch's maximum border distance. There may be small patches, i.e. patches with $r_{\max,i} < 1$ so that the full edge curve (3) won't fit on them. In nature, this corresponds to the situation in Fig. 1c. We solve this by setting the maximum snow height for small patches to $h_{\max,i} = r_{\max,i}$. Additionally, we need to rescale the edge distance according to $x' = x/r_{\max,i}$ so that $h_{\max,i}$ can be reached for at least one location within the patch.

The subdivision of the candidate locations into several snow patches S_i is similar to the patch subdivision for radiosity illumination as we also partition a scene for the computation of some resource balance, snow in our case. We now need to determine the amount of snow in each location dS when it comes to occlusions. Evidently, occluded faces should receive less snow than free surfaces. In eq. (3) we defined how to reduce the maximal snow height near patch edges, so we know how much snow is left for redistribution. Following our statistical model (1) we could again look at the distribution of the horizontal snow flake speeds after the first surface hit. But this may be arbitrarily complicated in general scene layouts. So we draw an approximate solution from a real world observation.

In Figure 5 a snow covered set of garden furniture is depicted. The snow on the table shows the typical edge profile. The feature essential to redistribution can be seen on the bench on the right. Apparently, the edge profile is approximately complementary to the profile above. This encourages our *principal redistribution scheme* for snow removed near edges:

If the full snow amount cannot settle at a location dS then the rest of it is passed downwards to the location lying directly below dS (in direction $\vec{f}$).

Figure 6 shows a sketch on how this shapes the snow cover. The definition above is sufficient in all cases, where there are snow locations below an edge region. If there are no snow locations directly below an edge we need an additional rule. In the close up inlay in Figure 8 you can see this kind of situation in the real world. One can see that the excess snow from above is transported horizontally and deposited there. Thus, we need to identify the regions that receive additional snow by horizontal transport. We call them inner edge regions. An inner edge is specified as a patch's border that has no lower neighbours (see Fig. 6). We denote the height of the additional amount of snow h_+. We define h_+ with the help of (3) in the new variable $\tilde{x}$, which is the distance to the inner edge

$$h_+(\tilde{x}) = 1 - h(\tilde{x}). \tag{5}$$

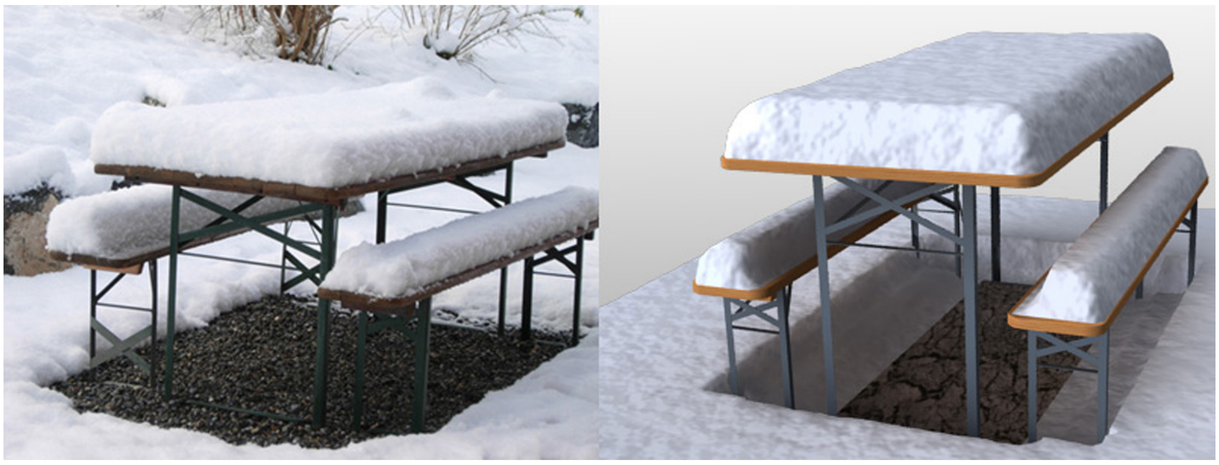

Figure 5: *Left: Snow covered garden furniture. Right: model with snow reproduced with our method. Grid resolution 150×104,* $h_{max} = 0.16$,*c=23*,$\alpha_m = 60^\circ$*, computation time ca. 370ms.*

The inner edge distance $\tilde{x}$ is stretched in the same way as the outer edge distance x of the outer edge that deposits its snow mass onto the inner edge. h_+ is only added for patches with inner edges. With this indirect modelling of horizontal transport we largely compensate for the snow mass loss encountered with the principal redistribution scheme only. In summary the total snow height H at a location j is computed as a function of x and $\tilde{x}$ (and possibly $r_{\max,i}$ for small patches)

$$H_j(x,\tilde{x}) = h_+(\tilde{x}) + h(x). \tag{6}$$

Obviously, our model characteristics (3) - (5) are simplified and not exhaustive for a comprehensive description of snow distribution. But they suffice to produce visually plausible snow covers as we will show in the next chapters. Moreover, we believe that our proposal (1) can serve as starting point for more precise snow cover approximations.

4. Implementation

Our implementation relies on a height span map [ON05]. The height span map consist of a square grid of transition lists. Each list is sorted by its transition height along the snowfall direction $\vec{f}$. There are two possible types of transitions: potential snow sites with their normal tilted less than 90° with respect to $\vec{f}$ i.e. air-object transitions seen from the sky, and object backsides, i.e. object-air transitions. In Figure 6 these transitions are indicated as red circles and squares. Each snow site transition holds its height, its snow height, the indices of its eight neighbouring indices, its snow patch index corresponding to i in eq. (4), and the discrete outer and inner edge distances corresponding to x and $\tilde{x}$ from eq. (3) and (5). There are three main steps in the implementation.

1. At first, we compute the all transitions in the height span map of a scene with a GPU-based depth peeler where the viewing volume corresponds to the scene's bounding box aligned along $\vec{f}$. One surface segment dS from (4) corresponds to one pixel in each depth peel layer. Depth-Peeling is both fast and easily adaptable for any snow cover resolution (see Fig. 6a). Typically, 3D - models contain unexpected face orientations contradicting their visual behaviour. This requires the application of two depth peel passes, one for front faces and one for back faces. We merge both depth texture groups in a way that every depth entry from a front side is followed by a backside. If this is not the case, i.e. if there are front faces without back faces or vice versa, we insert back (or front) faces directly below (above) it. This handled all mesh inconsistencies we found in our experiments.

2. In the second step, we group contingent snow patches as defined in (4). We achieve this by flood filling a patch from a random start site until all sites are assigned to a patch. On a square grid with a 4-neighbourhood the maximum surface tilt α from (4) can be translated to a maximal height difference $\Delta z_{\max}$ = cell size * $\tan\alpha$. As there might by several neighbouring snow sites in one grid position lying in this height range we choose to always prefer higher sites. This is efficient because the transitions lists are sorted by height. Moreover, during flood fill, inner edges, i.e. edges with a neighbour site larger than $\Delta z_{\max}$ are marked as seeds for further processing. The neighbour site above is also marked with a flag t which we will explain when computing the snow height.

3. In the third step, we compute outer and inner edge dis-

tances per patch. Again, we consider the four nearest neighbours and apply two distance transforms, i.e. we sequentially clip all sites in touch with the current border marking it with the current distance. For the outer distance transform we can trivially determine border contact by counting unprocessed neighbours (and by treating inner edge seeds as non-border). For the inner distance transform we use the seeds computed in the second step. We also save the maximum border distance for each patch. The regions marked with the flag t are grown to cover at least a region as broad as the edge curve's base.

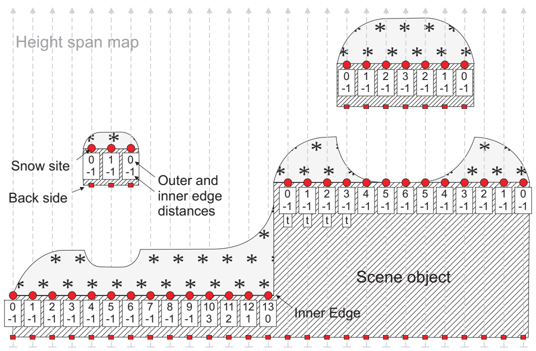

Figure 6: *Schematic view of the snow distribution algorithm. For clarity, the snow geometry is better resolved than the depicted height spans. The numbers in the squares denote inner and outer edge distances. The number -1 stands for undefined inner edges. For details see text.*

Then, we have provided all information to compute the final snow height. We simply need to iterate through each height span on every grid position. For one height span, we start on the top most position j and compute the snow height there according to (3). If $\tilde{x}$ is defined, i.e. if we are close to an inner edge, we increase the snow height by $h_+(\tilde{x})$ (cf. (5)). Then, if the maximum snow mass from the sky cannot be fully deposited i.e. if $h_j < h_{\max_i(j)}$ we transfer the excess to the next snow site below, on the same grid position (as in the principal redistribution scheme). Like this, we sweep through all snow sites until we reach the ground. Figure 6 shows a sketch of the resulting snow shape in a schematic scene. We could include a test here making sure that the snow height does not penetrate objects above a site. For rendering, the snow shapes defined by the snow height and the underlying objects are triangulated. We add gentle normal noise to increase realism when illuminating the scene. In very angular scenes or at low ground grid resolutions we finish snow distribution with a smoothing pass shifting each snow surface location towards the average position of its neighbours. For vertices at a snow patch's outer edge, we consider only those neighbours that also lie at the outer edge.

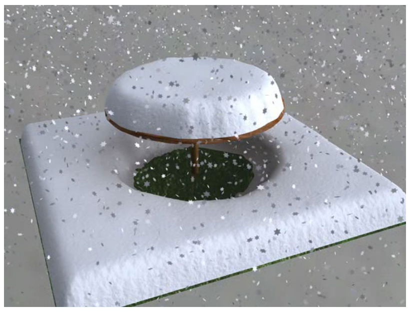

Figure 7: *Animated table. See movie in additional material. Computation time 90 ms for the generation of the initial mesh, all frames can be interpolated then. Grid resolution 75×75, $h_{max} = 0.15$,c=20,$\alpha_m = 60°$. Note [Tok06] Fig. 8 and [Fea00] Fig. 11 for comparison.*

5. Results and Rendering

All pictures in this paper were rendered with commercial 3D software if not indicated differently. All computation times refer to an AMD Athlon 64x2 Dual 3800+ 2.01 GHz CPU with 1 GB RAM and a NVIDIA GeForce 8800 GTS GPU. Maximal snow heights $h_{\max}$ are given in fractions of the scene height. Computation times refer to snow probability distribution calculation and mesh generation. To increase realism in the rendered pictures we added gentle noise to the snow height ($< 5\%$) and bump map texture.

In Figure 2 you can see an edge curve of a snow shape generated by our algorithm. Figure 5b shows garden furniture with snow cover. Note the realism of our reproduction of the vertical snow transport on the bench on the left. In the snow covered EG-Logo in Fig. 9 you can see both edge curves and vertical transport. In Fig. 7 one frame of a ten second snow cover build-up is shown. In Fig. 8 deposition close to inner edges are shown in comparison to a real world example. Figure 10 shows a complex virtual scenes covered in snow rendered in OpenGL. Figure 11 is composited scene with several snow covers generated separately.

6. Discussion and Future Work

As mentioned above, snow is a very complex material displaying a wide variety of shapes and structures. We showed that even a model as simple as ours can create convincing snow covers for scenes modelled without snow - if the model layout is chosen carefully. Our model's simplicity principally allows for faster computation than previously possible. In the scene shown in Figure (9) for example the creation of the snow model took about one and a half seconds with our conceptual C++-implementation. Comparison to previous models is difficult because either timing documentation is sparse (e.g. [Fea00]) or detail depth is much lower (e.g. [SEN08]) than in our model. The only work with detailed performance analysis [Tok06] unfortunately relies on

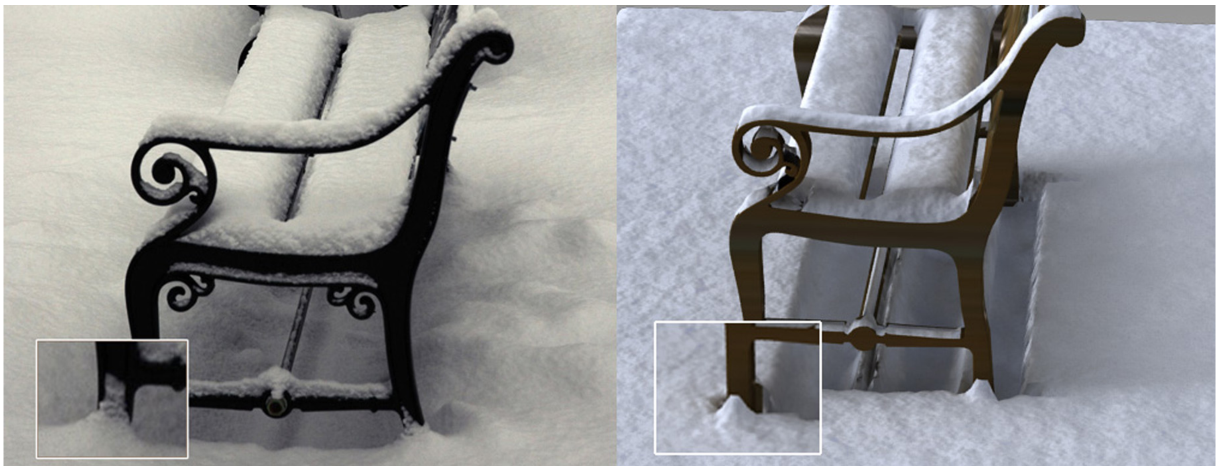

Figure 8: *Left: Snow covered garden furniture. Right: model with snow reproduced with our method. Grid resolution 187×300, $h_{max} = 0.065$, c=40, $\alpha_m = 80^\circ$, computation time 1.7 s.*

a manual preprocessing step. Though not directly comparable, our computation times are slightly faster. Moreover, for the animation of growing snow surfaces (see Fig. 7) our algorithm is principally faster as we don't need any stability tests. This improvement is the reason why our snow covers look more realistic than in [Tok06] Fig. 8 and [Fea00] Fig. 11. Our approach is ideal for animations: we compute a snow cover probability distribution once and can use it to determine a range of snow heights very fast and locally. With more optimization steps, we expect further performance gains. We only used efficient data structures and algorithms which were thoroughly investigated already.

A problem in our algorithm stems from the discrete scene sampling. Small segments in the scene e.g. the armrest in Fig. 8 might contain just a single line of snow with an outer border distance of zero at maximum so that no snow shape can be computed there. Two workarounds are possible: either the ground resolution must be as high as to cover each object in the scene with more than one snow site per grid direction. We found that best results can be achieved if one ground grid position corresponds to less than a square centimeter in the real world. The other work around could be a method we call small patch inflation. One would use the metaball technique as in [NIDN97] for all patches with $r_i \leq 1$. A randomly blurred implicit function would be sampled around all snow sites in the small patches. The snow surface would be rendered as isosurface then. But computation time would be significantly increased with this method, eventually outweighing timing gains due to lower resolution.

As shown above, our treatment of vertical snow redistribution is a coarse approximation. The snow redistributed at patch edges for example will not exactly settle below the edge in the real world as we assumed. Instead it will be diffused and spread depending on the vertical distance between the edge and the ground below. The same applies to inner edge redistribution. We could fit in these effects into our model by rescaling x and $\tilde{x}$ depending on the vertical distance to the edge where the snow comes from. Furthermore, in the real world, we observe additional wind driven redistribution. We could include a third distance similar to $\tilde{x}$ which was controlled by a wind field.

More subtle critical situations can arise with our inner edge redistribution method: e.g. an inner edge which is only accessible by a narrow cleft from above. In the real world, the cleft would be covered by snow and the inner edge would receive very little snow only. In our model all snow is transported to the bottom most inner edge. But when following our recommendation that one snow site covers about a square centimeter in the real world still a snow bridge would form if the cleft is smaller than a centimeter. We do not explicitly consider topological changes due to the snow cover neither. This would open up a huge range of new possibilities an algorithm would have to handle beyond the scope of our algorithm. Our algorithm aims to efficiently produce visually convincing snow covers in virtual scenes but not physical exactness. By further exploiting our statistical model snow bridges could be modelled as well: they will arise where the distance between two patches is small compared to the catchment area.

Two beautiful effects could also be included into our model. Single ungrouped snow sites at steep slopes could be group into thin-layer patches that could be painted with a semitransparent snow flake texture. And finally, overhangs observed after snowfall with large snowflakes could be modelled by adding an offset vector to the snow surface near edges. We leave these enhancements open to future works.

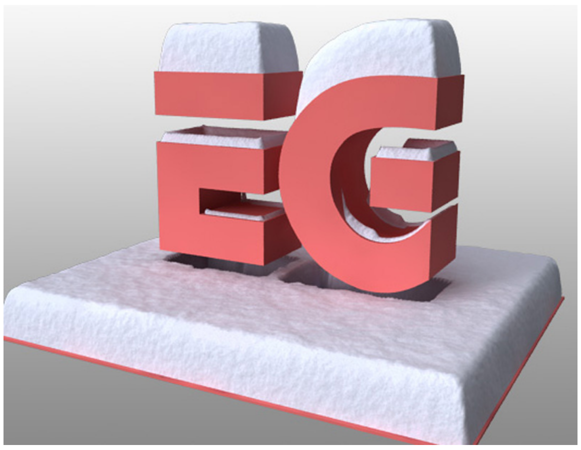

Figure 9: *Eurographics logo after snow fall. Grid resolution 100×118, $h_{max} = 0.18$,c=14,$\alpha_m = 70^\circ$, computation time 350 ms.*

7. Conclusion

We demonstrated a simple geometric algorithm to compute three dimensional snow covers of virtual scenes. Our algorithm covers a wide range of effects observed in snow distribution, although many effects necessarily remained ignored. We were able to create pictures close to photorealism and suitable for further usage in animations.

We thank David Körner for his support with rendering and 3D model creation.

References

[DL05] DEUSSEN O., LINTERMANN B.: *Digital Design of Nature - Computer Generated Plants and Organics*. Springer, 2005. 1

[Dud05] DUDASH B.: *Snow Accumulation*. Tech. rep., NVIDIA Corporation, 2005. 2

[FB07] FOLDES D., BENEŠ B.: Occlusion-based snow accumulation simulation. *VRIPHYS* (2007). 2

[Fea00] FEARING P.: Computer modelling of fallen snow. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2000), ACM Press/Addison-Wesley Publishing Co., pp. 37–46. 2, 6, 7

[FO02] FELDMAN B. E., O'BRIEN J. F.: Modeling the accumulation of wind-driven snow. In *SIGGRAPH '02: ACM SIGGRAPH 2002 conference abstracts and applications* (New York, NY, USA, 2002), ACM, pp. 218–218. 2

[GM81] GRAY D. M., MALE D. H.: *Handbook of snow: principles, processes, management and use*. Pergamon Press, 1981. 1

[HAH02] HAGLUND H., ANDERSSON M., HAST A.: Snow accumulation in real-time. *SIGGRAD* (2002). 2

[IGLF06] IRVING G., GUENDELMAN E., LOSASSO F., FEDKIW R.: Efficient simulation of large bodies of water by coupling two and three dimensional techniques. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers* (New York, NY, USA, 2006), ACM, pp. 805–811. 1

[LZK*04] LANGER M. S., ZHANG L., KLEIN A., BHATIA A., PEREIRA J., REKHI D.: A spectral-particle hybrid method for rendering falling snow. *Eurographics Symposium on Rendering* (2004). 2

[MC00] MURAOKA K., CHIBA N.: Visual simulation of snowfall, snow cover and snowmelt. *icpads 00* (2000), 187. 2

[Mel74] MELLOR M.: A review of basic snow mechanics. In *Proceedings of the Grindelwald Symposium, IAHS Publication No. 114* (1974). 2

[MMAL05] MOESLUND T., MADSEN C., AAGAARD M., LERCHE D.: Modeling falling and accumulating snow. *Vision, Video and Graphics* (2005). 2

[Nak54] NAKAYA U.: *Snow Crystals: Natural and Artificial*. Harvard University Press, 1954. 1, 2

[NIDN97] NISHITA T., IWASAKI H., DOBASHI Y., NAKAMAE E.: A modeling and rendering method for snow by using metaballs. *Computer Graphics Forum 16*, 3 (1997). (Proc. Eurographics'97) http://diglib.eg.org/EG/CGF/volume16/issue3/CGF172.html. 2, 7

[ON05] ONOUE K., NISHITA T.: An interactive deformation system for granular material. *Computer Graphics Forum 24*, 1 (2005), 51–60. 2, 5

[OS04] OHLSSON P., SEIPEL S.: Real-time rendering of accumulated snow. *SIGGRAD* (2004). 2

[PTS99] PREMOŽE S., THOMPSON W. B., SHIRLEY P.: Geospecific rendering of alpine terrain. In *In Eurographics Workshop on Rendering* (1999), pp. 107–118. 2

[SB08] SCHECHTER H., BRIDSON R.: Evolving sub-grid turbulence for smoke animation. In *Proceedings of the 2008 ACM/Eurographics Symposium on Computer Animation* (2008). 1

[SEN08] SALTVIK I., ELSTER A. C., NAGEL H. R.: Parallel methods for real-time visualization of snow. In *Applied Parallel Computing. State of the Art in Scientific Computing*. Springer-Verlag, 2008, pp. 218–227. 2, 6

[SOH99] SUMNER R. W., O'BRIEN J. F., HODGINS J. K.: Animating sand, mud, and snow. *Computer Graphics Forum 18*, 1 (1999), 17–26. 2

[Sto05] STOFFEL M.: *Numerical Modelling of Snow Using Finite Elements*. PhD thesis, ETH Zürich, 2005. 1, 2

[Tok06] TOKOI K.: Real-time rendering of accumulated snow. *In CGIV '06: Proceedings of the International Conference on Computer Graphics, Imaging and Visualisation* (2006), 310–316. 2, 6, 7

[WWXP06] WANG C., WANG Z., XIA T., PENG Q.: Real-time snowing simulation. *Vis. Comput. 22*, 5 (2006), 315–323. 2

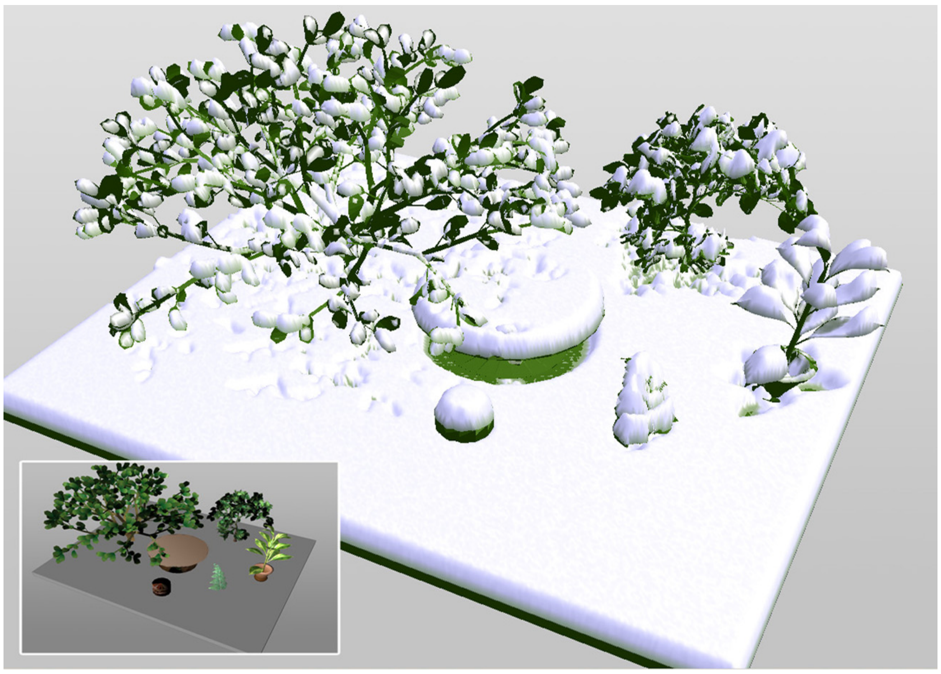

Figure 10: *Virtual scene covered with snow created with our method. Grid resolution 275×350, $h_{max} = 0.075$,c=50,$\alpha_m = 80^\circ$, computation time ca. 2.7 s, one smoothing pass, rendering in OpenGL with Alpha-Blending at low snow heights.*

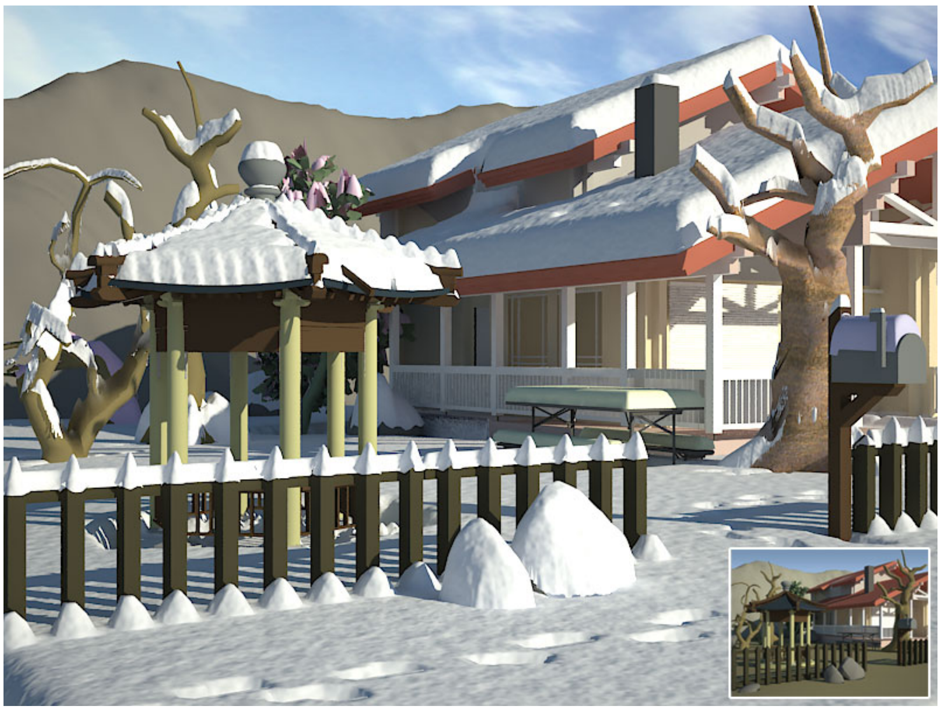

Figure 11: *Landscape in snow with a track of foot prints towards snow model optimization. Effective grid resolution 647×460, bulk computation time ca. 6.0 s, several snow covers computed separately.*