Computer Modelling Of Fallen Snow

Paul Fearing
University of British Columbia*

Figure 1: A sudden snowfall comes to the North Pole.

Abstract

In this paper, we present a new model of snow accumulation and stability for computer graphics. Our contribution is divided into two major components, each essential for modelling the appearance of a thick layer of snowfall on the ground.

Our *accumulation model* determines how much snow a particular surface receives, allowing for such phenomena as flake flutter, flake dusting and wind-blown snow. We compute snow accumulation by shooting particles upwards towards the sky, giving each source surface independent control over its own sampling density, accuracy and computation time. Importance ordering minimises sampling effort while maximising visual information, generating smoothly improving global results that can be interrupted at any point.

Once snow lands on the ground, our *stability model* moves material away from physically unstable areas in a series of small, simultaneous avalanches. We use a simple local stability test that handles very steep surfaces, obstacles, edges, and wind transit. Our stability algorithm also handles other materials, such as flour, sand, and flowing water.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modelling—Physically based modelling; J.2 [Physical Sciences and Engineering]: Earth and atmospheric sciences;

Keywords: snow, avalanches, stability, natural phenomena

*email: fearing@cs.ubc.ca

1 Introduction

One of nature's greatest beauties is the way fresh snow covers the world in a perfect blanket of crystalline white. It replaces sharp angles with gentle curves, and clings to surfaces to form ghostly silhouettes.

In many countries, snow is a common fact of life during the winter months. For example, January snow coverage in the Northern Hemisphere has ranged between 41.7 - 49.8 million square kilometres [17], or nearly half of the hemisphere's total land mass. A phenomenon that is so common and pervasive is clearly of interest and importance.

Despite the ubiquitous nature of snow, the entire season of winter has been almost completely ignored by computer graphics research and applications, with the exception of distant snow-capped mountains, and falling snowflakes. Without an automatic model of fallen snow, animators have so far relied upon intuition to produce snow-covered surfaces - an extremely tedious, time-consuming and potentially inaccurate task. A single tree might have a hundred branches, each with a complex drapery of snow, and each avalanching onto branches below, producing subtle second-order accumulation effects.

Besides the practicalities of research and application, there is another reason for investigating snowfall. Snow transforms commonplace scenes into fantastic wonderlands, greatly changing the appearance and mood of the landscape, allowing us to see familiar sights in a fresh, exciting way.

This paper presents a new method of snow pack modelling for computer graphics. We are primarily concerned with creating and simulating fallen snow at a scale where the thickness is clearly evident to the viewer. Our main emphasis is on a framework for efficiently handling large scenes with limited resources, and to a much lesser extent on a physically correct model of the snow itself. Snow is arguably one of the world's most complex naturally occurring substances, and accurate simulation is still a significant challenge to snow hydrologists and researchers.[1]

[1] We refer the reader to [9] or [1] for a discussion of the real substance.

Figure 2: A snow-covered gazebo with a hole in the roof. All snow was generated automatically, including snow on the mountains.

In order to generate images of a snowy world, we need to solve two major problems. Snow accumulation requires us to determine how much snow falls upon a scene, and where it accumulates. We simulate this with an adaptive particle/surface hybrid that addresses the proper allocation and conservation of snow mass around and under obstacles, the random nature of snowflake motion, and simple in-transit wind effects.

As snow accumulates, we compute snow stability in order to determine how much mass any particular surface can support. If not blocked by an obstacle, unstable surfaces release avalanches onto lower surfaces, also potentially covered in snow. We compute snow stability using a set of sequential local equations providing us with good results at a reasonable computational cost. Our approach allows us to simulate varying properties of both snow and like materials, as well as provide a simple model of mass transport due to wind.

Finally, we transform our model of accumulated, stable snow pack into a set of smoothly joining 3D surfaces that can be included in scenes or animations. During this step we can include bridging effects between nearby surfaces, as well as wind cornices. We augment our "thick" snow surfaces with flake dusting textures to provide extra noise and visual complexity.

Because of the sheer size and complexity of snowy scenes, our method is also inherently concerned with the practical issues of speed and control. Our primary contribution to this area is the counter-intuitive idea that snowflakes are shot *upwards* from individual surfaces, rather than dropped *downwards* from the sky. Giving individual surfaces control over their own "snowy destiny" allows us to prioritise computational effort on any number of criteria, including surface slope, area, distance to the camera, likelihood of interesting occlusions, or other measures of visual interest. Our algorithm provides a continuous, ever-improving result that can be terminated at any time, and still display the full snow depth.

As shown in Figure 3, our snow-adding algorithms are part of a larger pipeline involving a popular commercial animation package. Since the underlying scene remains unchanged, we retain the original lighting and animation and can rely upon strong commercial support for shader libraries and rendering. This makes it quite easy to add snow to a wide range of existing models and animations.

2 Related Work

Despite snow's common presence in many parts of the world, there has been little previous research towards a comprehensive model of snow for computer graphics.

Premoze et al. [16] generate realistic mountainous terrains that are likely the most convincing snow-covered scenes so far. Starting with a digital elevation model enhanced with an aerial photo, they use a detailed model of snow pack evolution to add zero-thickness patches of seasonal snow cover. The nature of the incoming satellite data restricts the technique to a scale much larger than our primary area of interest.

Muraoka et al. [13] simulate thick snow pack by dropping volume elements on the landscape, with provisions for snow evolution

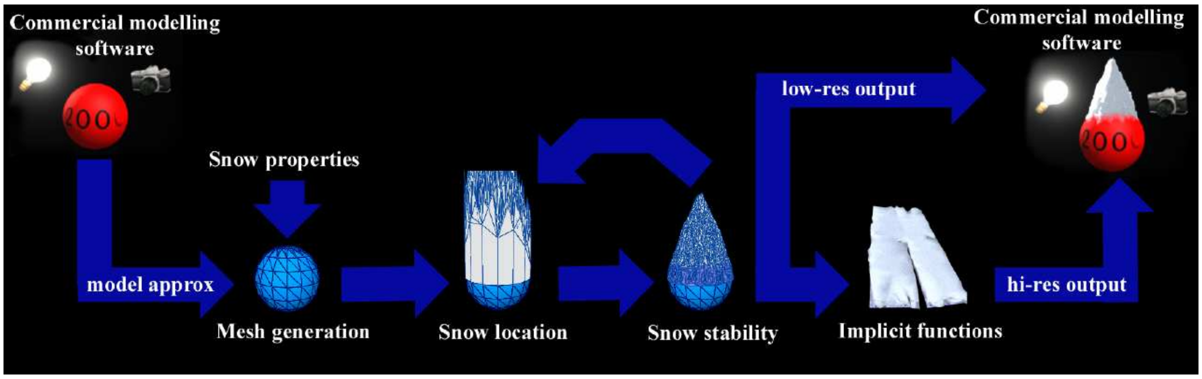

Figure 3: Overview of the snow pipeline. The underlying scene remains unchanged except for the inclusion of new snow surfaces.

[12]. Unfortunately, in order to cover the landscape with a computationally feasible number of particles, single-scale elements must be inflated to the point where they overwhelm underlying base surface detail.

Other work tangentially involving snow include Sims [20], and Shinya and Fournier [19], but both approaches are concerned only with falling and zero-thickness snow. Sumner et al. [21] simulate tracks in mud, sand, and snow using a regular height field and models of compression and erosion, but do not address snow accumulation. Nishita et. al [15] introduce a method of snow pack rendering based on multiple scattering of light within the snow volume. Snow surfaces were composed of individual metaballs placed by hand. Other work on snow illumination properties includes Hanrahan and Krueger [3] and Krueger [5].

Although not about snow, several other papers describe the motion of granular materials. Hsu and Wong [4] model zero-thickness dust accumulation with textures. Luciani et al. [8] introduce a multi-scale physical model for granular materials, designed to simulate such granular phenomena as piling, arching, and avalanching in the 2D plane. Li and Moshell [7] were responsible for a dynamic soil model on a constant regular grid, allowing for volume conservation, soil slippage, and manipulation of soil with a simulated bulldozer. Musgrave et al. [14] describe terrain generation, including an erosion and thermal weathering model that is quite applicable to snow stability.

3 Snow Accumulation

Peculiar to snow is the idea of “flake flutter”, where falling ice crystals are affected by crystal shape and atmospheric micro-turbulence. These local disturbances can prevent falling snow from descending in a straight line, instead allowing flakes to sidestep blocking obstacles and land underneath on surfaces that have no direct exposure to the sky. Thus, simulating and modelling an accumulation pattern is akin to raytracing for light, except that we are interested in *path* (instead of straight-line) visibility.

Where an obstacle, such as a porch or a bush, blocks the ground underneath, the flake flutter effect eventually produces an occlusion boundary between completely blocked and unblocked areas. An example of this can be seen in Figure 4(a), where snow accumulates well underneath the overhang of the bush. Over billions of flakes, these occlusion boundaries exhibit a smooth drop-off, where the shape of the curve and amount of snow under an object depends on the size, shape, and number of blocking occlusions, the closeness of the occlusion to the ground, and the magnitude of the fluttering effect.

For objects with many occluding components (such as a pine tree) the occlusion boundaries are still present, but are much less pronounced. Most falling snow accumulates on the uppermost layer of branches, but some accumulates on the next layer, and most lower branches and the ground get at least a small dusting. This contributes to the visual impression that snow is everywhere in a scene, and not just sitting on the uppermost surfaces exposed to the sky.

3.1 Computing the Snowfall Accumulation Pattern

Our goal is to generate an accumulation pattern for every surface in the model, where the amount of snow each surface receives is proportional to the occlusion factors described above.

Our approach is to allow launch sites on each surface to emit a series of particles aimed upwards towards a sky bounding plane. As particles flutter upwards, they are checked for intersection with intervening surfaces, where a “hit” indicates that a particle is somehow blocked, and cannot contribute snow to its source surface. A “miss” means that the particle made it through or around all blocking obstacles and reached the sky.

As particles reach or are blocked from the sky they slowly build a picture of a given launch site’s sky occlusion. Whenever a launch site has a sufficiently different sky occlusion from an adjacent neighbour, a new launch site is added at the perturbed midpoint to refine the transition. Likewise, launch sites can be merged whenever all surrounding neighbours have identical sky occlusions, usually in cases where sites are consistently confident that they are either completely exposed or completely occluded.

As soon as we have generated a mass accumulation picture that meets some resource criteria (compute time, number of samples, size of sample or some other importance-driven function) we can add an appropriate (and arbitrary) amount of snow. This generates a complete set of 3D snow surfaces that rise off the base model. Since the addition of a layer of blocking and obscuring snow changes the previously computed mass accumulation pattern, we can repeat the accumulation step as often desired, increasing accuracy at the cost of computation time.

3.2 Importance Ordering

The rationale for shooting upwards generally arises from the need for control: the idea that each individual surface can locally influence its resolution by deciding how many launch sites it needs, and how many particles each site should shoot. Since our sampling rate is orders of magnitude less complete than Nature’s, prioritising the few samples we do have allows us to make better use of them. This ensures that even the tiniest surface is guaranteed at least a rough estimate of snow accumulation. This is a major advantage over potential approaches that drop blobby particles, since small surfaces are often missed at the expense of covering large ones. Figure 16 shows

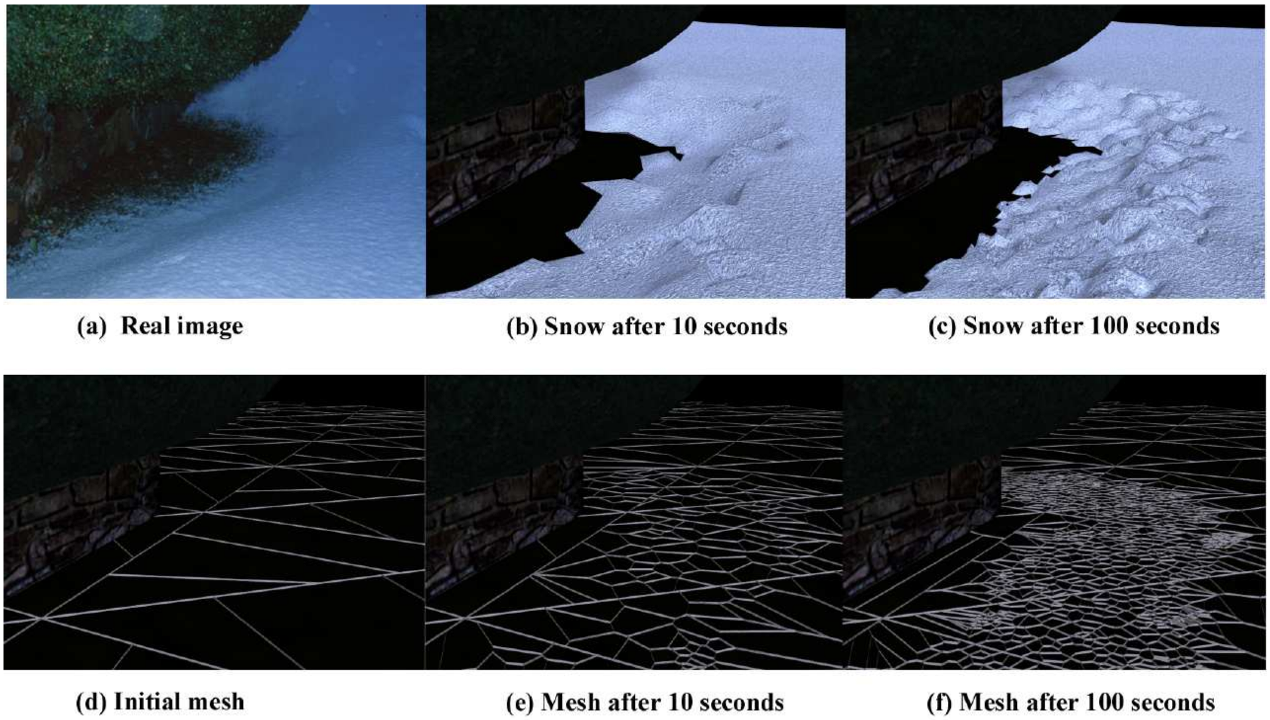

Figure 4: **(a)** A bush leaning out over a wall provides a real example of the flake-flutter phenomena. **(b)** After 10 seconds, importance ordering has found the general shape of the boundary. **(c)** After 100 seconds, the boundary shape is essentially the same as (b), due to the importance ordering of launch areas. For illustration, neither (b) nor (c) have been smoothed. **(d)** Initial meshing of a crude bush model. No measurement of the real bush was done. **(e)** The denser mesh reflects the more interesting areas. A significant amount of refinement occurs behind the bush and is not visible from this viewpoint. **(f)** The denser mesh after 100 seconds.

how our multi-scale approach covers individual blades of hay in the middle of a very large snowy field.

Each launch site is given an importance ordering used to determine order of site testing, determine the number of particles to shoot per site, and decide if more sites are needed nearby to improve the resolution. As long as the allocated time has not expired, the most important launch site shoots a small batch of particles, gets a new importance based on the results, and is placed back in sorted order. The importance ordering is a heuristic weighting based on the following factors:

- Completeness. Launch sites with no previous chances to shoot are more important than sites that have had at least one chance, ensuring a crude global approximation exists before any further refinement begins.
- Area. As the area of a launch site increases, particles from a single site will pass through less of the volume immediately overhead. To prevent missing occlusions, large areas may need more particles per launch site and more initial sites. Occlusion boundaries in large areas are more visually obvious, and so gain preferential allocation of new refinement sites.
- Neighbourhoods. If the particle hit percentage of two neighbouring sites is sufficiently different, it implies that there is a nearby obstacle causing some kind of occlusion boundary. Both sites gain importance, asking for more particles to improve knowledge of the shape, orientation and magnitude of the boundary. If the neighbours are sufficiently different and important, a new refinement site may be added to the perturbed midpoint. Likewise, launch sites that are the same as all nearby neighbours become less important, and may be candidates for removal.
- Effort. If all other factors are equal, launch sites should use approximately the same number of particles, aiming for consistency of confidence.
- Limits. The user can set several parameters that limit the approximate scale of the finest allowable increase in resolution. This prevents launch sites from increasing indefinitely along very complex occlusion boundaries. If all sites have been resolved to this limit, the phase can terminate early.
- Steepness. Very steep launch sites are swept of what little snow they accumulate; in most scenes, these avalanches are negligible compared to accumulations on nearby stable sites.
- Camera. When optionally enabled, sites closer to the camera receive more particles, greater refinement, and improved accuracy at the cost of imposed view dependence.
- User. Importance ordering allows users to arbitrarily tag surfaces as being "boring" - useful for ignoring areas that will eventually be occluded or matted out.

We defer the reader to [1] for the actual parameterised importance weighting and further discussion of each factor. The important idea is that some launch sites get priority access to a limited sampling budget, based on criteria important to the user for a particular scene.

Figure 4(a) shows the occlusion boundary under a real snow-covered bush, illustrating the type of visual effects we want sampling to determine. After 10 seconds, the importance ordering has found the boundary, and generated an initial approximation. Spending an additional 90 seconds results in more subtle improvement, refining launch sites of less visual interest. Background unoccluded areas are of very low importance, and so undergo almost no improvement.

3.3 Launch Site Meshing

Launch site surfaces are represented as triangles, generated from the original (potentially non-polygonal) base models. Once snow has been generated, the polygonal approximation of the underlying model is discarded, allowing snow to accumulate on the original, unchanged base scene.

All upwards-facing triangles in the approximation of the underlying model are initially allocated at least one launch site. Additional launch sites are allocated based on the importance ordering of the surface, user-set resolution parameters, and the magnitude of the flake-flutter.

In order to properly allocate snow, each launch site must be responsible for some non-overlapping portion of the surface, ideally the area immediately surrounding the sample point. We have chosen a strategy based upon Voronoi diagrams, although there are numerous other valid meshing possibilities. Launch sites are connected in a constrained Delaunay triangulation, where each launch site is responsible for its own immediately surrounding Voronoi area, clipped to the edge of the triangle for maximal surface independence. Advantages of this approach include fast point-in-area tests and neighbour location, and the ability to quickly generate triangulations for intersection testing.

Figure 4 (d) shows an example of a sparse initial mesh undergoing the addition of more and more launch sites, shown in Figures 4 (e) and (f). Note how neighbouring constrained-Voronoi areas vary in size at the transition zones, and mostly minimise extreme angles. In practice, many surfaces are small and isolated (such as the brush and pine needles in Figure 1), and meshes are reduced to the trivial case of one or two samples in a triangle. Significant meshing occurs on large, connected surfaces, such as the ground.

Launch sites and their associated meshes are additionally divided into *edge groups*, which are isolated world objects, projected into the XY plane, bordered by the XY silhouette edges. Edge groups are used primarily for avalanche resolution, denoting sharp boundaries where snow may slide off from one edge group to another. Projecting into XY implies that launch sites can only be placed on surfaces with an angle of repose of $[0..90)^\circ$. Since edge group silhouettes are not necessarily convex, we must do some additional processing to "break" constrained Delaunay neighbour links that cross a silhouette boundary or a hole in the mesh. A single edge group may also be arbitrarily broken into smaller edge groups, although this is inefficient since moving snow across group boundaries is more expensive than moving snow within the same edge group. Figure 5 shows how a sphere is converted into an edge group.

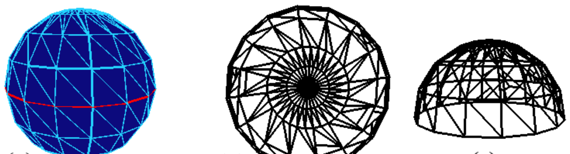

Figure 5: An isolated object (a), bordered by XY silhouette edges (in red) forms an edge group - top view (b), side view (c).

Our particular meshing strategy means that we have trouble with certain types of connected models that overlap in Z, such as a helix. However, this can be fixed by either splitting the model's natural object hierarchy, or increasing the number of edge groups, ultimately reaching the level of a group per polygon, if needed. Figure 6 shows an overlapping Z model that our meshing algorithm considers hard. Note that although the knot was split into 200 edge groups, boundaries between the groups are not visible in the final result.

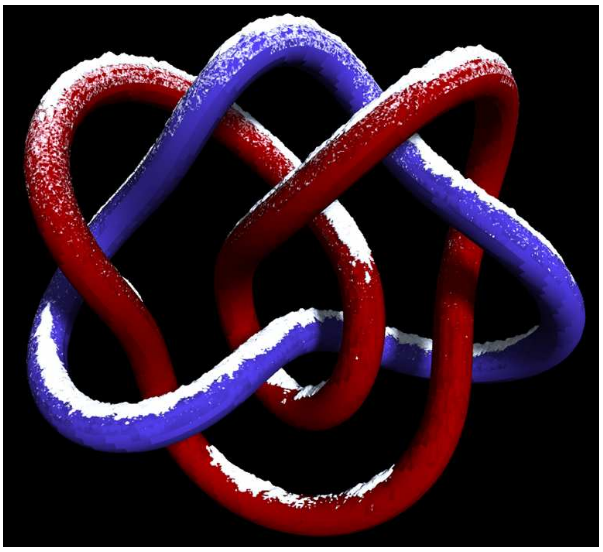

Figure 6: An object our meshing strategy considers "hard". Knot model courtesy of [18].

3.4 Locating Particles in the Sky

When a launch site reaches the head of the importance queue, it shoots a batch of particles towards the sky. Batch size is user definable, but generally within the order of 10-15 flakes. Particles originate from the launch site's snow surface, potentially reaching the sky plane unimpeded and contributing to the growth of the parent. We use a simple bucketing and filtering scheme to allocate the successful flakes to the total mass of the sky's available snow, while ensuring that small local areas of sky do not over-contribute. This is important, since the number of particles hitting any particular area of the sky may vary dramatically depending on the complexity of the underlying surfaces. We must ensure that a large concentration of flakes (say, directly above a tree), draws the same total snow as would the sky above a sparse flat surface. Furthermore, importance ordering implies that not all launch sites shoot the same number of particles.

We divide the sky into a grid of constant size buckets. When a flake reaches the sky successfully, we spread its representative area (defined as the launch site's projected area divided by the number of flakes in the current batch) across one or more buckets, as shown in Figure 7.

When the snow accumulation phase finishes, all sky buckets are allocated some mass based upon the arbitrary depth of snow desired. Each bucket b computes a mass per area value, based on available mass of b and the summation of all representative flake areas extending into b. An individual launch site l then receives new mass proportional to the summation of the representative area of all flakes belonging to l that hit b. A single launch site may receive snow from multiple buckets. Flake area filtering is done at the end of the accumulation phase, when a given launch site cannot change in area due to added or removed refinement sites.

Since a launch site's accumulation pattern may change with the addition of blocking snow, it is sometimes useful to split the desired snow depth up and run the accumulation phase more than once. Depending on the time allocated for each phase, lower-importance launch sites may not get a chance to shoot particles every pass. To

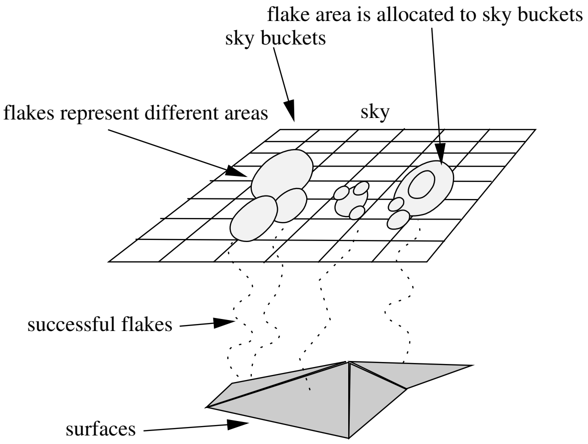

Figure 7: Allocating flake area to sky buckets.

allow fair mass allocation to those launch sites, we keep flake information in the sky until replaced by a "fresher" set of shot particles from a new pass.

The allocation of snow mass to sky buckets is usually constant, although interesting effects can be obtained by multiplying bucket mass by an input image. Figure 8 shows a scene where the sky generates very uneven amounts of snow.

Figure 8: Non-constant allocation of snow mass to sky bucketing can be used to "write" the SIGGRAPH 2000 logo with snow

3.5 Snowflake Motion

We simulate snowflake motion with a series of straight-line vectors approximating a curved path, where vector length and end position are determined with a random walk process based upon a circle of radius f_r, and Z step resolution is influenced by the importance ordering. At each step, the value of f_r is randomly chosen from a normal distribution. As f_r approaches zero, flakes duplicate vertical raycasting, producing no partial occlusion. As f_r increases, the "area of effect" of a flake widens, generally blurring occlusion boundaries and making it less obvious where bumps and depressions came from.

In practice, it is hard to match flake-flutter parameters with observed real scenes. We currently compare a grid of generated images to find the parameters that best match the shape of boundaries found in a real scene.

3.6 Determining Particle/Surface Intersection

In order to find particle/surface intersections, we allocate surfaces into a regular grid of XY buckets. Within each bucket, we compute the minimum and maximum Z values of the surface as it passes through the bucket bounding box. We then insert all Z ranges into a per-box range tree [23]. For a tree containing n ranges, it takes $O(log^2 n)$ per insert and delete, and $O(log^2 n + k)$ to return a list of the k elements that overlap the Z query range. During the accumulation phase, rebucketing is only needed upon completion, when snow mass is added. During stability, rebucketing is done more often, although with a considerable lag for efficiency reasons.

3.7 Surface Construction

After snow allocation, each launch site is elevated by recently accumulated snow mass divided by the current launch site area. The polygonal top snow bounding surface is then the constrained-Delaunay triangulation of elevated launch sites, with corner vertices set to the minimum of adjacent neighbours. Additional vertical planes are included around edge group boundaries to close the surface down to the base plane.

3.8 Flake Dusting

In many instances, accumulated snow is not thick enough to completely obscure the underlying surface, appearing instead as a light "dusting"of flakes. This phenomena often occurs in areas of low snowfall, high instability, or on surfaces with microtexture bumps, such as tree bark. Since it is not practical to model dusting as thick 3D objects, we use already-computed snow occlusion percentages to generate procedural noise textures of the appropriate averaged dusting density. Dusting textures are semi-transparent, textured polygons oriented to float slightly in front of the original model. Figure 9 compares the texture dusting of a (slightly tilted) real and a computer generated sign. Figure 6 shows an example of the transition between thick surfaces and flake dusting textures.

In a view-dependent scene, flake dusting can be used to replace sufficiently thin and distant snow layers with a white texture, reducing the polygon count.

4 Snow Stability

The snow stability phase of the algorithm is responsible for redistributing recently accumulated snow mass into a configuration that is stable, according to some very simple surface and snow properties. It can be run at intermittent times as computational power and desired accuracy permit, usually immediately after snow accumulation.

All launch sites are initially sorted by absolute Z height plus accumulation, and placed in a list of unresolved sites u_1. The list is examined in decreasing Z order, immediately resolving unstable launch sites as they are discovered. The resolution of a single launch site s may affect a number of nearby neighbours: lower sites may receive new snow from s, while the loss of snow from s may create unstable angles with previously stable higher neighbours. Affected samples also include sites receiving edge-transit snow from s, or sites newly created to improve resolution.

If not there already, all launch sites affected by s, including s, are placed in a new sorted list u_2. At the completion of an entire pass through u_1, the list is destroyed and replaced with u_2, and the entire pass is repeated until termination.

The length of u_1 is not guaranteed to decrease on each pass, and in fact may increase, or undergo large fluctuations. Consider a large amount of very unstable snow on a wide flat surface. On the first pass, the vast majority of interior samples are considered stable, since they are at the same height as their neighbours. The band of instability exists only at the edges, where unsupported snow avalanches off into the void. As edge sites lose mass, adjacent interior neighbours are affected, and the area of instability widens. Fortunately, the erosion of snow from the edges towards the centre is very physically plausible.

Figure 9: (a) A real sign covered with real snow. (b) A computer generated sign covered with computer generated snow. Note how dusting density increases near the top and edges in both models.

4.1 Angle of Repose

Despite the wide range of physical factors influencing real snow, for simplicity we base our stability test mainly on the angle of repose (AOR) of a particular snow type. The AOR measures the static friction of a pile of granular material, and is one of the major parameters influencing our scene. It can range [6] [9] from near $90°$ in fresh dendritic snow to $15°$ in extreme slush conditions.

For a given type of snow, we use a transition curve that models the probability of stability over a range of angles around the AOR. Increasing the width of the transition curve gives a stability solution with bumpier surfaces and increased variation at snow boundary edges near the critical angle. A narrow curve generates smoother surfaces with less variation.

The AOR is based on the relative heights of accumulated snow, and not on the fixed angle of launch sites on the underlying surface. As snow drains from one launch site to another, the AOR changes continually. This means that launch sites on very steep surfaces may still support snow if the AOR of neighbouring sites is low enough, possibly because snow is blocked from moving away.

Figure13 shows an example of this using water (AOR = $0°$) filling a fountain basin. The basin sides are too steep to support water, so mass avalanches towards the basin bottom. As the basin fills, this downward movement is blocked by the rising water level. Eventually, the basin fills to the brim, leaving a stable flat surface supported by the steep sides of the bowl.

4.2 Stability Test

The actual stability test for a single iteration on launch site s can be described as follows:

1. compute AOR between s and all neighbours n_i lower than s
2. for each i with an AOR too steep to support snow, perform an obstacle test between s and n_i
   - i. if there is a non-snow obstacle in the way, the avalanche is blocked, and the neighbour n_i is ignored.
   - ii. if there is a vertical snow surface (an edge group boundary) in the way, there an interpenetrating surface carrying snow between s and n_i, so the avalanche is also blocked.
   - iii. if there is a non-vertical snow surface in the way, there is an interpenetrating surface B between s and n_i, where the interpenetrating surface could potentially receive the snow destined for n_i. Replace n_i with the closest launch site on B.
3. evenly shift snow from s to all neighbours n_i still in contention, until at least one neighbour becomes stable.
4. repeat steps 1 to 3 until all there are no unstable neighbours left, or s is bare of snow.

Figure 10 illustrates some of the obstacle cases.

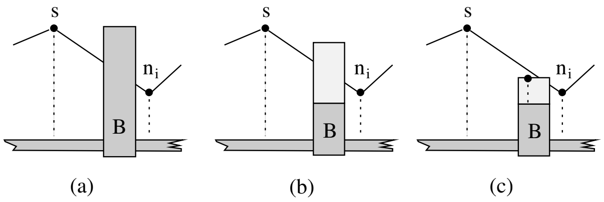

Figure 10: (a), (b), (c) illustrate stability test obstacle cases i, ii, and iii respectively

The obstacle test (step 2) checks to make sure that avalanche motion is not blocked by intervening surfaces or snow belonging to other objects. If an obstacle is found, snow is blocked and forced to pile up unless there is an alternative escape direction or snow rises above the intervening obstacle. Figure 13 shows how blocked water rises above the level of the basin sides, transferring to the top of the basin edge, eventually overflowing into the next basin.

Step 2 is expensive. Practically, we achieve large speedups by reducing the frequency of this step - from every test, to every pass, to once per stability phase, with corresponding decreases in accuracy. The most infrequent testing is usually sufficient for models where there is little inter-object penetration, although some blocking due to rising snow will be missed. Figure 1 was computed using the fastest method. Figure 16, containing thousands of interpenetrating and closely spaced grass blades, was computed using the slowest method.

Any time there is a non-snow obstacle between two adjacent neighbours, we can optionally improve the way snow builds up against the obstacle by adding refinement launch sites just before the intersection point.

4.3 Moving Snow over Edges

If an unstable launch site has no downhill neighbours, it is next to an edge. Before snow cascades over an edge into the air, we perform an intersection test with a very short vector oriented in the direction of avalanche motion. If an intersection is found, then some surface or nearby snow is sufficiently close to the avalanche origin to block movement. Blocked avalanches continue to accumulate until the origin launch site has enough snow to pass over the obstacle. If no intersection is found, the avalanche heads over the edge and is approximated as a few (usually < 5) avalanche particles moving on a simple projectile trajectory. Avalanche particles are tracked downwards, bouncing off surfaces until reaching a surface supporting launch sites. If the edge is on a shared boundary with an adjacent edge group, the particles end up "hopping" to the adjacent group via very short projectile motion.

When an avalanche particle comes to rest, it contributes its snow load to the nearest launch site on the destination surface. Depending on user-set parameters, new launch sites may be created if existing launch sites are not dense enough to capture the pattern of falling snow.

4.4 Stability Termination Criteria

A single pass of the stability algorithm reaches completion when it runs out of time, when the unresolved list u_1 becomes empty, or when all avalanches in the last pass moved only a very small amount of snow.

In most scenes, the first few passes through u_1 resolve a majority of the unstable snow, with subsequent passes handling smaller and smaller avalanches. Forced early termination may leave unstable areas, but all launch sites will usually have avalanched at least once. Our multi-pass approach avoids driving a large wave of snow downwards in a single pass, which leads to chaotic results on early termination.

If the stability phase completes before the alloted time expires, we re-run the entire phase to compensate for some speed-accuracy tradeoffs, such as missed obstacle testing, and lag in the rebucketing of changing snow surfaces. The extra phase usually fixes a few missed sites and completes immediately.

Figure 11: Covering a hydrant with low AOR snow.

5 Implicit Functions

Importance-ordering accumulation algorithms are surface-based, implying that snow can only accumulate on supporting objects. To allow for unsupported snow, such as gap bridging, edge bulges and wind cornices, we perform an additional (optional) conversion step using implicit functions. Figure 2 shows an example where snow on many closely-spaced pine needles has formed unsupported bridges and clumps.

Each snow volume is converted into one of several different implicit function types, as shown in Figure 12. Generator functions do not radiate uniformly. The one-sided "edge" function allows bulging and cornice formation, where size, bulging and direction are based upon wind velocity. The limited "top" function blends with snow directly above the generator surface, but does not blend much with adjacent neighbours. The resulting isosurface is polygonalized in $O(n^2)$ space [22].

In order to reduce blending discontinuities and apparent mass inflation at function boundaries, we use known adjacency information to shrink and clip implicit functions so that the isosurface is coincident with the polygonal top surface. A small variable-radius line generator function blends cracks between adjacent functions, and smoothes over sharp creases in the snow. Our method is not entirely satisfactory, since surface cracks often remain visible - however, they are often minimised sufficiently to be destroyed during mesh reduction [2] after polygonization.

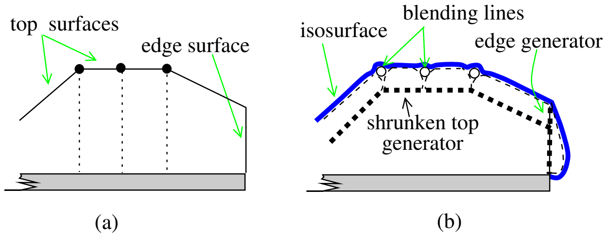

Figure 12: (a) Side view of adjacent snow volumes. (b) Side view of adjacent top and edge generator functions, with crack-filling blending lines.

Implicit functions potentially allow us to add animal tracks, wind ripples, and other patterns to snow surfaces by "stamping" the snow surface with appropriately scaled negative functions.

By interrupting the pipeline before the implicit function step in Section 5, we obtain polygonal results with no bridging or smoothing effects and a much lower polygon count. These compact intermediate results are appropriate for scene setup and real-time viewing, and may actually be sufficient for the final image. Figures 4, 13 and 16 were computed without the smoothing step. As well, intermediate polygonal results can be used as the underlying model for a completely new snow accumulation run, producing the effect of true snow layers.

6 Rain and Wind

By setting AOR = 0° and flake-flutter $f_r = 0$ the basic snow algorithm can also simulate the accumulation of water, from the sky or elsewhere. Figure 13 shows an example of an empty fountain slowly filling up with water. Only the patch of sky shown as a red square has any mass to contribute, approximating how water appears at a spout, fills the first basin, and overflows to lower basins.

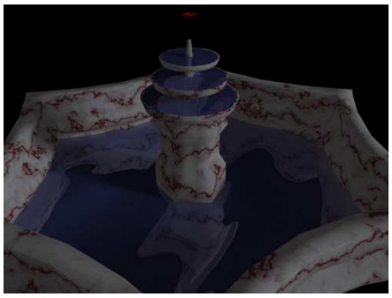

Figure 13: Snow stability algorithms can also be used to simulate water accumulation. Water from the red patch fills the first basin before overflowing into subsequent basins.

Wind is a major factor in the large-scale transport of snow, producing some very compelling and interesting effects. Although we cannot claim to duplicate these effects, we at least have a framework for simple wind phenomena in both snow accumulation and snow stability phases.

During snow accumulation, wind influence is easily included by modifying a flake's direction and distance by a velocity vector. Wind velocity vectors can be approximated with a constant direction, or much more accurately computed offline. The foreground haystack in Figure 16 shows the asymmetrical accumulation effects of a very slight breeze to the right, where the wind influence is globally constant.

During stability, we widen our single-site stability test to include neighbours that are within $90°$ of the downwind direction. Snow transport is then dependent on the neighbour's angle with respect to the local wind vector, the duration of the wind influence, and the carrying capacity of a given wind velocity, based on [10]. The instability vector is moved according to the rules of Section 4.2, including obstacle testing.

We use a simple heuristic to compensate for the different number of times each launch site may be stability tested. All launch sites compute a flux maximum that is reduced and moved over some small number of stability trials. Unfortunately, if the stability phase is terminated early, some areas may not get a chance to move all allowable wind transport snow. Figure 14 shows an example of wind and stability effects using a simple, globally constant wind vector.

Figure 14: (a) The initial scene without wind. (b) A globally constant wind blows the snow against the wall. Much of the snow has blown completely away.

7 Validation

Validation of snow-covered scenes is hard, in that snow observed outdoors is the result of uncontrollable and unknown environmental factors. Creating artificial snow is beyond our capabilities as a graphics lab, so instead we restrict validation to observation, asking the question: "does our algorithm produce phenomena and/or effects that are observable in nature?"

However, we were able to perform a few simple experiments to show that our snow stability algorithms are at least plausible. We substituted sifted flour for snow, to improve controllability and show that our algorithms work for materials other than snow. Figure 15 shows a side-by-side comparison of real and computer generated flour scenes. Figure 9 shows an additional side-by-side validation image of flake dusting.

8 Future Work

Our initial focus was on a framework for snow generation, and as a result we ignored, simplified, and actively avoided many extremely important physical properties and effects, including snow compression and packing, layers, slab avalanches, snow creep, snow pack metamorphosis, melting, and solar influence.

Other priorities include improving the overall smoothness of the final results. Our sampling method is very noisy, mainly due to the (relatively) tiny number of flakes used to extrapolate snow depth.

Figure 15: A real flour-covered scene (a) and a computer generated scene (b) compared to show that our stability algorithms are at least plausible. Our experimental setup was fairly ad-hoc: despite our best efforts, flour was distributed unevenly around the base of the real sphere.

Additionally, avalanching real snow distributes snow in a much wider and more complex cloud that we currently model with our few particles, leading to snow stalagmite artifacts, such as those near the foreground wall in Figure 1. Although we are able to artificially enforce surface smoothing, we have not done so in this paper.

Timing results are not fully applicable to our importance ordering scheme, as models are usually allocated a running time convenient to the user. However, the timing bottleneck of snow as a useful effect is the rendering phase, which is outside the scope of our current work. Large models such as Figures 1 and 2 were given overnight for snow accumulation, yet required weeks to raytrace animations of several hundred frames. Rendering is aggravated by aliasing in moving scenes - such as the distant, tiny, white snowpatches resting on distant, tiny, dark needles shown in Figure 1. We are interested in physically realistic, multi-resolution snow shaders or rendering models that are fast and accurate.

9 Conclusions

This paper describes a new algorithm for the creation of snow-covered models, using a novel particle location scheme that allows surfaces to independently control sampling effort needed to determine accumulation. Separability of surface accumulation produces many useful side effects, including importance ordering, adaptive refinement, smooth degradation upon early termination, and greater control of the final result. Our accumulation algorithm allows us simulate effects such as accumulation under obstacles, flake dusting, wind, falling rain, and "snow-writing".

We have also presented a simple model of snow stability that handles avalanches, edge-transit snow, obstacles supporting and blocking snow, materials other than snow, and mass transport due to wind. Additional features of the approach include support for snow bridges, cornices and various levels of model detail. Integration with commercial software allows us to snow upon existing models in a variety of formats, providing greater flexibility, power, and ease of use. Finally, we have shown that our approach is able to handle large, complex outdoor scenes consisting of hundreds of thousands of surfaces.

It is our hope that this work will open up an entire new season to computer graphics, and will stimulate other researchers to explore the natural, glorious beauties of winter.

10 Acknowledgements

Alain Fournier provided guidance and the haystack model, while colleagues and the anonymous reviewers provided many helpful suggestions. Most of base models were provided courtesy of Platinum Pictures.

Figure 16: A snowy scene inspired by Monet [11]. This model shows the results of omitting the final implicit smoothing phase. Haystack models courtesy of Alain Fournier.

References

[1] Paul Fearing. *The Computer Modelling of Fallen Snow*. PhD thesis, Dept. of Computer Science, University of British Columbia, July 2000.

[2] Michael Garland and Paul Heckbert. Surface Simplification Using Quadric Error Metrics. *SIGGRAPH 97 Conference Proceedings*, pages 209–216, August 1997.

[3] Pat Hanrahan and Wolfgang Krueger. Reflection From Layered Surfaces Due To Subsurface Scattering. *Computer Graphics (SIGGRAPH 93 Conference Proceedings)*, 27:165–174, August 1993.

[4] Siu-chi Hsu and Tien-tsin Wong. Simulating Dust Accumulation. *IEEE Computer Graphics and Applications*, 15(1):18–22, January 1995.

[5] Wolfgang Krueger. Intensity Fluctuations And Natural Texturing. *Computer Graphics (SIGGRAPH 88 Conference Proceedings)*, 22(4):213–220, August 1988.

[6] Daisuke Kuroiwa, Yukiko Mizuno, and Masao Takeuchi. Micrometrical Properties Of Snow. In *International Conference on Low Temperature Science (Physics of Snow and Ice)*, volume 1, Part II, pages 722–751. Institute for Low Temperature Science, Aug 1966.

[7] Xin Li and Michael Moshell. Modeling Soil: Realtime Dynamic Models For Soil Slippage And Manipulation. *SIGGRAPH 93 Conference Proceedings*, 27:361–368, August 1993.

[8] A. Luciani, A. Habibi, and E. Manzotti. A Multi-Scale Physical Model Of Granular Materials. In *Proceedings of Graphics Interface*, pages 136–137. Canadian Information Processing Society, 1995.

[9] David McClung and Peter Schaerer. *The Avalanche Handbook*. The Mountaineers, Seattle, Washington, 1993.

[10] Malcolm Mellor. Engineering Properties Of Snow. *Journal of Glaciology*, 19(81):15–66, 1977.

[11] Claude Monet. *Wheatstacks, Snow Effect, Morning*. Painting: oil on canvas, J. Paul Getty Museum, Los Angeles, 1891.

[12] K. Muraoka and N. Chiba. A Visual Simulation Of Melting Snow. *The Journal of the Institute of Image Electronics Engineers of Japan*, 27(4):327–338, 1998.

[13] K. Muraoka, N. Chiba, and I. Ohtawara. Snowfall Model For Simulating Close Views Of Snowy Landscapes. *The Journal of the Institute of Television Engineers of Japan*, 49(10):1252–1258, 1995.

[14] F.K. Musgrave, C.E Kolb, and R.S. Mace. The Synthesis And Rendering Of Eroded Fractal Terrains. *Computer Graphics (SIGGRAPH 89 Conference Proceedings)*, 23(3):41–50, July 1989.

[15] T. Nishita, H. Iwasaki, Y. Dobashi, and E. Nakamei. A Modeling And Rendering Method For Snow By Using Metaballs. In *Proc. EUROGRAPHICS*, volume 16. European Association for Computer Graphics, 1997.

[16] S. Premoze, W. Thompson, and P. Shirley. Geospecific Rendering Of Alpine Terrain. In *Eurographics Rendering Workshop*. European Association for Computer Graphics, June 1999.

[17] D.A Robinson. *Northern Hemisphere Snow Cover Charts*. National Snow and Ice Data Center, http://www-nsidc.colorado.edu/NSIDC/EDUCATION/SNOW/snow_Robinson.html, as of April 10, 2000.

[18] Robert G. Scharein. *Interactive Topological Drawing*. PhD thesis, Department of Computer Science, The University of British Columbia, 1998.

[19] Mikio Shinya and Alain Fournier. Stochastic Motion – Motion Under The Influence Of Wind. In *Proc. EUROGRAPHICS*, pages 119–128. European Association for Computer Graphics, 1992.

[20] Karl Sims. Particle Animation And Rendering Using Data Parallel Computation. *Computer Graphics (SIGGRAPH 90 Conference Proceedings)*, 24(4):405–413, August 1990.

[21] R. Sumner, J. O'Brien, and J. Hodgins. Animating Sand, Mud and Snow. In *Proceedings of Graphics Interface*, pages 125–132. Canadian Information Processing Society, 1998.

[22] Alan Watt and Mark Watt. *Advanced Animation and Rendering Techniques*. Addison-Wesley Publishing, Don Mills, Ontario, 1992.

[23] D.E. Willand. New Data Structures For Orthogonal Queries. *SIAM Journal of Computing*, 14(1):232–253, 1985.

Figure 17: Another view of Figure 1.

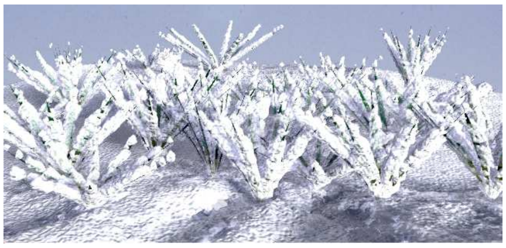

Figure 18: Snow covered brush.