DOI: 10.1111/cgf.13379

EUROGRAPHICS 2018 / D. Gutierrez and A. Sheffer
(Guest Editors)

Volume 37 (2018), Number 2

Interactive Generation of Time-evolving, Snow-Covered Landscapes with Avalanches

G. Cordonnier[1], P. Ecormier[2], E. Galin[3], J. Gain[4], B. Benes[5], M.-P. Cani[2]

[1]Univ. Grenoble Alpes, Inria [2]Ecole Polytechnique [3]LIRIS-CNRS Université de Lyon
[4]University of Cape Town [5]Purdue University

Abstract

We introduce a novel method for interactive generation of visually consistent, snow-covered landscapes and provide control of their dynamic evolution over time. Our main contribution is the real-time phenomenological simulation of avalanches and other user-guided events, such as tracks left by Nordic skiing, which can be applied to interactively sculpt the landscape. The terrain is modeled as a height field with additional layers for stable, compacted, unstable, and powdery snow, which behave in combination as a semi-viscous fluid. We incorporate the impact of several phenomena, including sunlight, temperature, prevailing wind direction, and skiing activities. The snow evolution includes snow-melt and snow-drift, which affect stability of the snow mass and the probability of avalanches. A user can shape landscapes and their evolution either with a variety of interactive brushes, or by prescribing events along a winter season time-line. Our optimized GPU-implementation allows interactive updates of snow type and depth across a large ($10 \times 10\,km$) terrain, including real-time avalanches, making this suitable for visual assets in computer games. We evaluate our method through perceptual comparison against exiting methods and real snow-depth data.

CCS Concepts

•***Computing methodologies*** → ***Shape modeling;*** •***Human-centered computing*** → *Interaction techniques;*

1. Introduction

Mountainous snow-covered landscapes are among the most visually-arresting vistas. In nature, snow coverage depends on altitude, but also a host of other phenomena, such as snow melting more on sun-facing slopes, snow shifted by the wind as channeled by topography, avalanches scouring some of the steepest slopes, and human activities, such as skiing, which leave visually-prominent imprints. Such snow-covered landscapes are heavily used in animated films and computer games, where their static portrayal provides a compelling backdrop, while dynamics elements (such as ski tracks and avalanches) serve a storytelling function, but a manual modeling process predominates. The challenge in instead generating these phenomena through simulation, lies is achieving both plausible results and the efficiency necessary for control.

While previous approaches in Computer Graphics do address snow coverage, we are not aware of any that account for the variety of contributing phenomena. Furthermore, most of the focus has been at the extremes: either fine-scale (e.g., objects draped with snow [Fea00], ice crystal formation [KL03]) or large-scale effects (e.g., snow-cover simulation based on ambient occlusion [FB07]). Medium-scale landscapes with, in particular, the effective user control of a visually-consistent snow layer have yet to be investigated.

We tackle the problem of developing an interactive design tool for consistent snow-covered landscapes at medium scale (from 1×1 km to 10×10 km in extent, modeled using a $1\,024 \times 1\,024$ grid) – a resolution fine enough to capture snow-drift, avalanche and ski-track effects, but broad enough to encompass an expansive vista. The main challenge lies in supporting interactive creation of both static and time-evolving landscapes, while maintaining the consistency of snow coverage throughout.

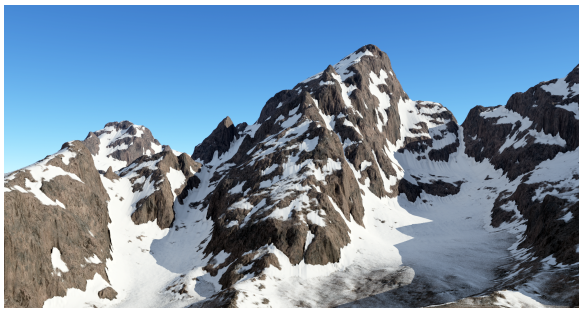

Figure 1: *A snow-covered landscape computed with our method.*

The key observation of our work is that avalanches and other phenomena such as snow-fall, snow-melt, snow-drift and Nordic

Figure 2: *Our framework generates snow-covered landscapes with triggered avalanches. The input scene (left) is enhanced with user-painted snow (middle) and avalanches are triggered, leading to a complete scene with snow beneath the mountains (right).*

skiing have significant visual impact on snow-covered landscapes. They should form part of the designer's tool-set, since they literally sculpt the landscape. While existing methods allow for the controllable placement of static snow cover, avalanches and snow-drift cannot be achieved as easily because they involve dynamic phenomena that *reallocate* snow mass. In contrast, these dynamic phenomena – optionally guided by the user – are at the heart of our modeling pipeline.

The input to our method is an elevation map overlaid with several qualitatively-different layers of snow (compacted, stable, unstable, and powdery), which are initialized by the user and set to evolve over time. For each time-step (e.g., one day), the snow cover is updated as a consequence of various phenomena. First, we account for evolution in the local snow composition, such as a shift from stable to unstable snow with warmer temperatures. Second, a number of external events act on the snow layers. As examples: wind shifts powdery snow, avalanches may occur in steep areas with unstable snow, and skiing both compacts powdery snow and potentially triggers avalanches. These events sculpt the landscape, leading to the formation of characteristic features, such as overhangs on crest lines caused by snow-drift.

Thanks to our efficient GPU-accelerated implementation of both stochastic phenomenological simulation for general events and fluid flow specifically for avalanches, this time-evolution is computed at interactive rates. To improve visual feedback, despite the day long time-step, avalanches and dynamic ski tracks are visualized in real-time using a temporal zoom mechanic.

A user designs the landscape and controls scenarios for its temporal evolution using brushes: any point on the simulation time-line can be selected and tools applied to adjust the snow layers, either globally or locally. The probability of avalanches and the area of ski activity can also be altered. The user can thus rapidly explore design alternatives within the parameter space of events. Figure 2 shows a typical use-case, with a terrain and its initial simulated snow coverage (left), with snow depth then sculpted by a designer (middle), and finally the consequences of an avalanche (right).

We claim the following technical contributions: first, a novel interactive framework for modeling snow-covered countryside built on the interactive GPU simulation of Poisson-based stochastic events, acting on a layered terrain model. Second, an efficient method for visually simulating avalanches that combines viscous fluid and granular material behavior in a balance dictated by external conditions. This is seamlessly integrated with the stochastic simulation framework. Third, a simple, yet accurate method for efficiently generating ski-tracks, which accounts for human impact on the uppermost snow layer and provides compelling visual detail.

2. Related work

Existing methods for modeling snow cover fall into three broad categories: particle-based, physically-based heat transfer, and procedural surface displacement.

Particle-based techniques infer a distribution of surface snow by computing the trajectory of snow particles as they are blown by the wind and accumulate on objects. In early work, Nishita *et al.* [NIDN97] reconstruct a smooth implicit snow surface by treating snow particles as skeletal point elements. Fearing [Fea00] introduced probabilistic snow cover by sampling a surface with snow particles and backtracking their trajectory to source clouds while accounting for collision, thereby arriving at a particle likelihood. Subsequent work incorporates wind perturbation when tracing the inverse trajectory, using Navier-Stokes [FO02, MaMAL05], Boltzmann approaches [WWXP06] or even cellular automata-based approximations [MC95], with possible acceleration through parallelization [SEN07]. Hinks *et al.* [HM09] generate realistic accumulation patterns by tracing snow-carrying particles in a dynamic wind-field and define the volume of the snow using an implicit-surface model. Stomakhin *et al.* [SSC*13] developed a semi-implicit Lagrangian method for simulating the complex behavior of snow under different environmental conditions.

There are also benefits in a hybrid treatment of snow that separates static structure represented by voxels [SKIT15] or a height-field [DGP16], from dynamic elements implemented with particles. This strategy even enables interaction between snow and dynamic objects [DGP16]. However, a common limitation of particle-based approaches is their inability to scale beyond small scenes with an upper limit of about 100×100m.

Physically-based heat transfer methods account for thermal interactions between objects with a view to simulating snow accumulation, ice formation and snow melt as influenced by environmental conditions. These principles have been applied by Muraoka *et al.* [MC00] with snowfall simulated using vortex fields and snow melt using heat conduction. Maréchal *et al.* [MGG*10] go further by simulating snowfall and conductive, convective and radiative thermal transfers using a finite volume method over a voxel grid.

This accounts for the complexity of phase changes where snow melts to become water or water freezes into ice. Unfortunately, the realism of heat transfer comes at a high computational cost, since it requires a memory-heavy volumetric representation of the scene and complex radiative transfer equations.

Procedural surface displacement methods generate plausible snow cover by offsetting the surface of objects in a scene using a phenomenological approach. An initial strategy is to characterize the height of accumulated snow using local accessibility and occlusion characteristics [PTS99]. This can be extended to large-scale terrains by linking snow buildup to ambient occlusion and snow dissipation to direct illumination [Tok06, FB07, RLD15] or wind-drift approximated by directionnal occlusion [MT10]. Another acceleration strategy is to rely on multi-mapping and treat nearby objects (with a displacement map) and distant objects (with a volumetric texture map) differently [CSLW03]. Height span maps, borrowed from the context of granular material dynamics [SOH99], have also proven effective in modeling real-time snow accumulation with the incorporation of phenomenologically-inspired statistics [FG09, FG11]. In recent work, Grosbellet *et al.* [GPG*16] provide a general architecture for detailing small but complex scenes, by allowing objects in the environment (such as lampposts, trees, and fountains) to affect scalar parameter fields (such as temperature, fallen-leaf density and humidity), which ultimately dictate decoration of the objects themselves (with snow, ice and fallen leaves).

A stochastic simulation of events that modify the state of a layered terrain was introduced by Cordonnier *et al.* [CGG*17] to model the interactions between vegetation and erosion. They treat events as generators of a series of simulation steps running locally and in isolation over the terrain until the associated phenomenon (such as a rock fall) terminates. Applying this to our goals would be impossible, since we require a framework where real-time catastrophic events can cascade, such as a skier triggering an avalanche, both displayed in real time and thus running simultaneously. To this end, we use Poisson processes to trigger stochastic event-steps that represent an increment of a given phenomenon over the whole terrain. This, in combination with a GPU implementation, enables us to achieve real-time results.

3. Overview

Our method enables users to interactively design static and dynamic medium-scale snow-covered landscapes (typically, $1-10$ km on a side), at a ground-plane and temporal sampling resolution that is sufficient (typically, $1-10$ m per cell and one day per time step). In particular, we want to capture characteristic snow-field effects, such as snow cornices on the leeward side of mountain crests, buildup at the base of slopes scoured by avalanches, and ski-tracks that weave downhill. Lastly, to enhance the user experience and improve control, we want to visualize fast dynamic events such as avalanches or new ski tracks generated at the speed a user expects from real life, even if the remainder of the simulation runs at a faster pace.

Our method generates snow cover for a scene, represented by a height-field grid with additional layers for compacted, stable, unstable and powdery snow, by drawing from a set of concurrent phenomena, including precipitation (snowing in our case), snow melt and wind transport (Figure 3), which interact, mutually but indirectly, through the shared snow layers.

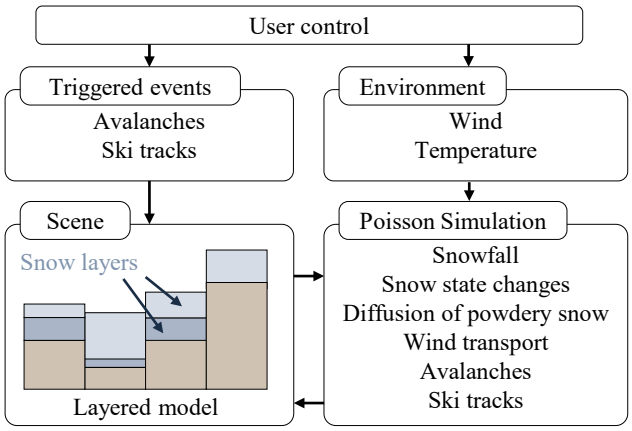

Figure 3: *System overview: The user can define the environmental conditions that influence the snow simulation, or directly and interactively trigger events such as avalanches or new ski tracks. These are used in our unified stochastic Poisson simulation. The output is a static or dynamic snow-covered landscape.*

These phenomena, in turn, are strongly influenced by time-linked environmental conditions, like temperature, prevailing wind, and sunlight intensity. Avalanches and skiing represent a class of complex limited-duration phenomena that require special treatment, particularly if they are to be incorporated as dynamic, real-time effects in games and virtual environments. There are intricate evolving interactions between the layered scene, the evolution of snow cover over time, and the environment. One causal chain might involve wind shifting powdery and unstable snow across a crest onto a sheer slope, setting up ideal conditions for an avalanche to be triggered by an oblivious skier. To achieve such intricate simulation at interactive rates, the key idea is to use Poisson processes for interweaving incremental steps of the various events to be simulated, often at quite different temporal scales (presented in Section 3.1 and Figure 4).

The user can direct the simulation in several ways: by defining a temporal scenario for changes in environmental conditions, by directly painting height changes into the snow layers, by interactively exploring the phenomena parameters, or by triggering avalanches and skiing. In the interests of interactive performance, all components of the framework are implemented on a GPU.

3.1. Simulation method

The scene model consists of a regular grid. Each cell contains a stack of height values representing a set of ordered layers: bedrock (the initial static height-field input), compacted snow (an icelike layer subjected to pressure from above), stable snow (a cohesive layer bound strongly to the terrain), unstable snow (a weak layer susceptible to slippage when perturbed), and powdery snow (an aerated layer with little cohesion subject to constant small spills).

Environmental conditions: An interactive session starts with a pre-computation step, where the environmental conditions are used to compute initial snow cover (Section 4). They include temperature (computed from altitude and illumination) and wind (computed from altitude and wind speed at sea level). These phenomena are only considered when the simulation is launched, or if the user decides to change environmental conditions. Results are stored as static layers over the terrain using a scalar field for temperature and a vector field for wind speed, and used, together with the snow cover layers, to compute the local effect of the different run-time events, all of them being applied over the whole terrain.

Poisson Stochastic Simulation: At runtime, the phenomena of interest require different time scales: for instance, the typical timescale of melting snow is far longer than required for a running avalanche. To achieve this, a Poisson process is associated with each phenomenon, which is thus expressed through a series of individual stochastic events, applied on the whole terrain at a given mean-frequency f_e. We use a random variable t_e to represent the time at which the next event of type e is to be triggered. The variable t_e follows an exponential distribution defined by the probability density function:

$$\mathcal{P}_e(x) = f_e e^{-f_e x} \text{ if } x \geq 0 \qquad \mathcal{P}_e(x) = 0 \text{ otherwise.}$$

Since the set of times t_e for the different event types are independent stochastic variables, the probability of the next event being e is:

$$P\Big(e \mid t_e = \min(t_1, \cdots, t_n)\Big) = f_e / \sum_{i=1}^{n} f_i \tag{1}$$

The simulation algorithm thus runs as follows: At each simulation step, the system randomly selects which event is next according to the distribution of probabilities in Eqn (1), and triggers it. The global simulation time is then increased by the mean time-step associated with the current phenomenon.

Such event-based time-steps seamlessly support *temporal zooming* when high-frequency events are triggered. For example avalanches and ski-tracks, are assigned a much higher frequency than other events, enabling them to be perceived as dynamic phenomena.

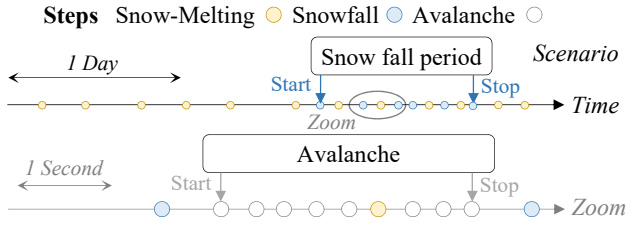

Figure 4: *Temporal development of snowfall, snow melting, and an avalanche. Each event is handled by a Poisson process, some, such as the snowfall and avalanches, are preceded and followed by start and stop events. Avalanche steps have a high frequency and an entire avalanche may take place between two lower frequency events (such as snowfall steps).*

3.2. Categories of events

We split the different phenomena into a series of stochastic, infinitesimal *events*, each applied to the whole terrain (with, however, a different local effect depending on environmental conditions and snow layers). In addition to simulation step events, some phenomena such as snowfall require a start event followed by given a number of steps. We use two categories of events:

Snow cover evolution encompasses phenomena that take place over a longer period through a slow series of stochastic events (Section 5). For instance, *Snow-Melting* and *Wind-Transport* (snow transported elsewhere by the wind) are two phenomena that are always active and that take place through a series of *Snow-Melting-Step* events and *Wind-Transport-Step* events, respectively, each occurring twice a day on average. Another phenomenon is *Snowfall*, triggered by a *Snowfall-Start* event occurring once per week on average, followed by a series of *Snowfall-Step* events that are triggered twice a day over a period of a few days. Each *Step* event updates both snow distribution and snow stability.

Interactive phenomena include avalanches and ski tracks in our framework. Both take place over a short period (Section 6). In addition to being stochastically called at runtime (e.g., once a week on average for avalanches on a one square kilometer terrain, and once an hour for ski tracks), events such as avalanches and ski tracks can be directly triggered by the user. This is achieved by increasing the frequency of the associated *Avalanche-Start* (respectively *Ski-Track-Start*) event, after creating favorable conditions at the user selected location, until the requested event is launched. The same method is used to enforce catastrophic events, such as ski-tracks triggering immediate avalanches when they cross unstable snow layers. While an avalanche (respectively a Ski-Track) phenomenon is active, *Avalanche-Step* (respectively *Ski-Track-Step*) events take place every $0.1s$ on average, which enables the user to see the dynamic phenomenon taking place at its natural (real-time) speed.

4. Environmental conditions

Snow coverage changes depending on the locally-varying environmental conditions, among which the most important are temperature (as affected by sunlight) and wind. These conditions depend primarily on terrain topography with the snow layers having negligible impact. Because of this, we can precompute these conditions and update them only when the user changes the terrain or an input parameter (such as wind direction or sea-level temperature).

4.1. Temperature

In our simulations, temperature is a key environmental input that depends primarily on altitude A and sunlight exposure I. Strictly speaking, total altitude should be the combination of terrain altitude B and snow thickness D. However, since the snow contribution is small ($D \ll B$) we use an approximation $A = B$ that allows precomputation. Temperature T is calculated as:

$$T = T_0 + k_t A + k_i I.$$

T_0 is the temperature at sea level, $k_t = -0.01°C\,m^{-1}$ is the per meter decrease in temperature with altitude, and $k_i = 10°C$ is the temperature increase due to 24-hours of direct sunlight.

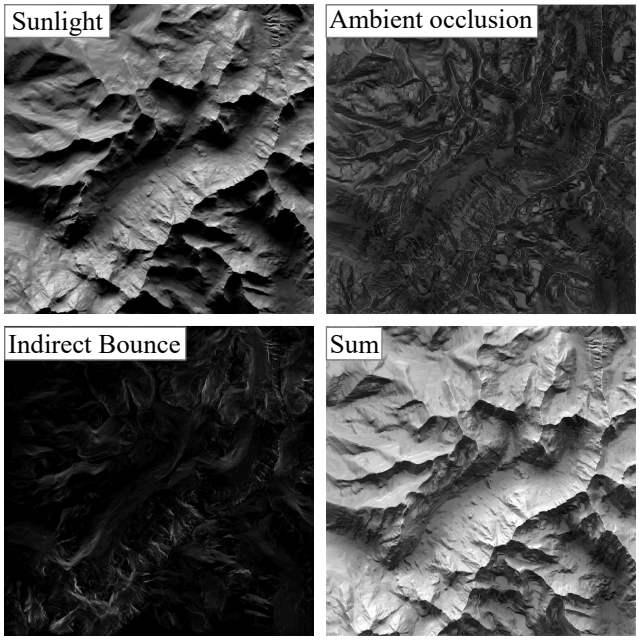

Figure 5: *Sun exposure is computed by summing the direct sunlight, ambient occlusion, and one bounce of the indirect sunlight.*

The calculation of sun exposure is more involved, since we factor in direct illumination potentially blocked by topography, ambient occlusion resulting from scattering due to clouds [FB07,MGG*10], and a single ray bounce [RGS09] accounting for the high reflectivity of snow. Total exposure (Figure 5) is then a sum of direct I_{sun}, ambient I_{sky}, and indirect sunlight I_{ind}: $I = I_{sun} + I_{sky} + I_{ind}$.

Direct sunlight for a point $\mathbf{p}$ on the terrain is measured as the proportion of sun exposure during a day and is given by:

$$I_{sun} = k_{sun} \sum_h s_h \, \mathbf{n} \cdot \mathbf{i}_h,$$

where k_{sun} is sun intensity, s_h indicates topographic shadowing of an incident ray $\mathbf{i}_h$ from the sun's position at hour h (factoring in latitude and time of year [CGG*17]), and $\mathbf{n}$ is the terrain normal at the point $\mathbf{p}$.

For ambient exposure, we make the assumption that ambient light I_{sky} is rotation invariant. We sample directions $\theta_i \in [0, 2\pi]$ around a terrain point $\mathbf{p}$ and obtain the maximal angle ϕ_i between the tangent to the terrain and the line between $\mathbf{p}$ and the highest visible point on the terrain in this direction (Figure 6, left). Then for an infinitesimal angle $\mathrm{d}\theta$ around θ_i, the ambient exposure is:

$$\mathrm{d}I_{sky,i} = k_{sky} \int_{x=\phi_i}^{\pi/2} \cos(x)\sin(x)\,\mathrm{d}\theta\,\mathrm{d}x = k_{sky}\frac{cos^2(\phi_i)}{2}\mathrm{d}\theta.$$

Total ambient sunlight at a point is the integral of the term above, which we discretize as a sum:

$$I_{sky} = \sum_{i\in[1,N]} \frac{2\pi}{N}\frac{\mathrm{d}I_{sky,i}}{\mathrm{d}\theta} = k_{sky}\frac{\pi}{N}\sum_{i\in[1,N]} cos^2(\phi_i).$$

We evaluate indirect sunlight at a point $\mathbf{p}$ (the receiver) using a polar coordinate sampling of the nearby direct sunlight over random directions θ_i and distances $\{r_0, \ldots, r_j\} \in [0, r_{max}]$.For each such sample $\mathbf{p}_{i,j}$ (the sender) we define an angle $\phi_{r_{i,j}}$ between the receiver's normal and the sampling direction $(\mathbf{p}_{i,j} - \mathbf{p})$. Conversely, $\phi_{s_{i,j}}$ is the angle between sender's normal and the inverse sampling direction (Figure 6, right). Indirect sun exposure is calculated as:

$$I_{ind}(\mathbf{p}) = k_{ind} \sum_i \sum_j A_{i,j}\, I_{sun}(\mathbf{p}_{i,j}) \frac{\cos(\phi_{s_{i,j}})\cos(\phi_{r_{i,j}})}{d_{i,j}^2},$$

where $A_{i,j} = 2\pi(r_{j+1/2}^2 - r_{j-1/2}^2)/N$ is the sampling area, for N angular samples, $r_{j+1/2} = (r_j + r_{j+1})/2$, and $d_{i,j}$ is the distance between $\mathbf{p}$ and $\mathbf{p}_{i,j}$.

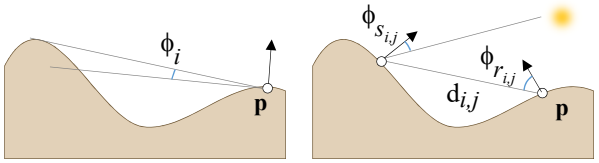

Figure 6: *Terms used in the calculation of indirect sun exposure.*

4.2. Wind

In a snow-cover simulation, wind is both critical for its role in snow transport, and complex since it depends on topography. As a first approximation, we treat wind as a 2D velocity field that sits atop the terrain, and is initialized according to a globally uniform dominant 2D wind vector $\mathbf{W}_d$. This field is then enhanced by taking into account the effects of altitude, local slope direction, and wind shadows. Here again, we set A=B, allowing for precomputation.

First, we account for *Venturi* effects that accelerate wind at high altitudes, by scaling wind velocity with height:

$$\mathbf{W}_{venturi} = (1 + k_{venturi} A)\mathbf{W}_d$$

While this linearization of the Venturi effect is not physically accurate it has the advantages of being efficient and easily controllable. Moreover, in our experiments the resulting snow coverage was not visually distinguishable from results obtained with more accurate simulation.

Second, we redirect wind according to terrain slope. Let $\mathbf{n}_{xy}$ be the horizontal component of the normal vector to the surface. We define $\mathbf{n}_{xy}^{\perp}$ as the 2D vector obtained after 90° rotation of $\mathbf{n}_{xy}$ in the direction of $\mathbf{W}_{venturi}$ ($\mathbf{W}_{venturi} \cdot \mathbf{n}_{xy}^{\perp} \geq 0$). When the terrain is almost flat, $\|\mathbf{n}_{xy}\|$ is small and the wind direction does not change. Otherwise, the wind tends to align with $\mathbf{n}_{xy}^{\perp}$ as captured by the equation:

$$\mathbf{W} = \mathbf{W}_{venturi}\left(1 - \|\mathbf{n}_{xy}\|\right) + k_{terrain}\|\mathbf{W}_{venturi}\|\mathbf{n}_{xy}^{\perp}. \qquad (2)$$

Finally, wind shadows form leeward of crests and ridge lines and this leads to the characteristic build-up of snow cornices. To achieve this, we assume that wind shadows are binary (wind at full strength or no wind at all), and we create a *wind-effect surface* representing the lowest altitude above which the wind blows at full strength.

The horizontal wind velocity for this layer is set to $\mathbf{W}$ (Equation 2). The altitude of the wind-effect surface W_z is related to the

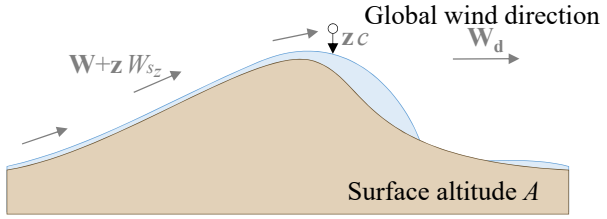

Figure 7: *While the wind-effect surface is in contact with the ground ($W_z = A$), its vertical speed W_{s_z} is forced to follow the terrain gradient. Otherwise, W_{s_z} decreases by a constant value c, which makes the wind-effect surface form a parabolic shape.*

vertical wind speed W_{s_z} as follows:

$$\begin{aligned} \mathbf{W} \cdot \nabla W_{s_z} &= -c && \text{if } W_z > A \\ W_{s_z} &= \mathbf{W} \cdot \nabla A && \text{otherwise,} \end{aligned}$$

where c is the constant by which the vertical wind speed decreases when the wind-effect surface is strictly above ground level (Figure 7).

The value of c is set experimentally to $0.7\,m\,s^{-2}$, and controls the extent of the wind-shadow cap. The smaller this value, the bigger the cap. The layer's altitude is then derived from:

$$\mathbf{W} \cdot \nabla W_z = W_{s_z}.$$

The equation is undefined for $\mathbf{W} = \mathbf{0}$, in which case we set $W_{s_z} = A$. Leeward of ridge lines, the gradient of the elevation and the wind act in opposite directions creating a negative vertical speed that forces the wind-effect layer to slowly decrease, generating a parabolic wind-shadow cap. Windward, the wind is pushed toward the surface, where W_{s_z} is directed by the 2D gradient of A.

Both altitude W_z and vertical speed W_{s_z} are computed using a Gauss-Seidel scheme. They are initially set to $W_z = A$ and $W_{s_z} = \mathbf{W} \cdot \nabla A$. Iteration proceeds in a black-and-white checkerboard fashion. On odd iterations, all white-assigned grid cells are updated with respect to black-assigned cells, as follows: first, W_{s_z} is calculated, independently from W_z; then W_z is updated based on the new value of W_{s_z}; and finally, W_{s_z} is corrected if $W_z \leq A$. On even iterations the black cells are re-evaluated.

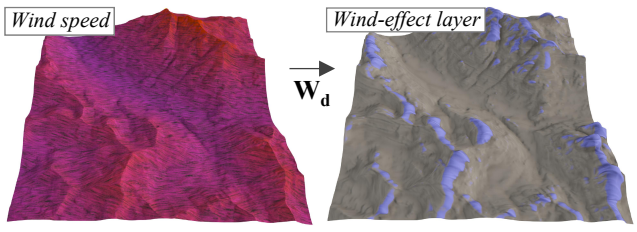

Figure 8: *Wind speed increases with altitude – shown as color ranging from blue to red (left) with dark lines indicating the local wind directions; the wind-effect surface hugs the ground except leeward of ridges, where caps (in blue) are formed (right).*

5. Snow cover

Snow is rarely homogeneous in either structure or composition. To express this we partition snow cover (D) into four layers: compacted (C), stable (S), unstable (U), and powdery (P) snow. The distinction between the latter two is necessary because powdery snow often trickles downwards immediately after deposition, while unstable snow shifts only with an avalanche event. The unstable layer serves as a proxy for the interleaving of brittle and strong layers in real snow that are likely to trigger avalanches.

Key to of our framework is tracking the evolution of these snow layers over time, under the action of precipitation, melting, diffusion, and wind shifts (see Figure 9).

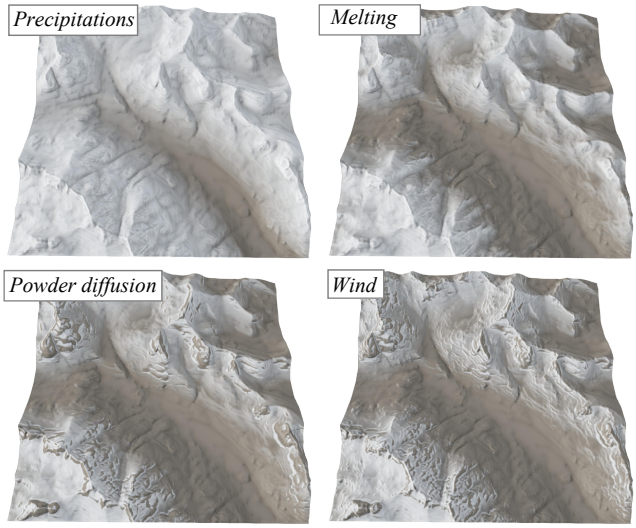

Figure 9: *Effects of different phenomena on snow cover.*

5.1. Snowfall

Snowfall is a blanketing event that influences snow cover by adding to the uppermost layers. We use a mean triggering interval of one week and a mean duration of three days as defaults. Apart from these, the user can also adjust snowfall strength k_{snow}.

The Foehn effect allows a straightforward relationship between snowfall and altitude:

$$D(t + \delta t_p) = D(t) + \delta t_p \, k_{snow} \max(0, A - A_{0\,precip}),$$

where $A_{0\,precip}$ is the altitude at which rainfall transitions to snowfall, and $\delta t_p = 1$ day is the precipitation time step. Snowfall is initially divided into powdery and normal snow, with the latter assigned to S and U layers after landing. First, some of the precipitated snow is converted into powdery snow (i.e., snow that cannot bond to bare slopes or the existing snow layer). The proportion of snow that becomes powdery is $x_P = k_{powdery}(\|\nabla A\| - s_c)$ (clamped between 0 and 1), where $k_{powdery}$ is a user defined constant that scales the slope influence, $\|\nabla A\|$ is the norm of the gradient of snow surface altitude, and s_c is a critical slope beyond which the snow

cannot fall. Based on the assumption that this critical slope depends on the inertial temperature of the underlying ground, the critical slope increases when the temperature decreases:

$$s_c = s_{c_0} + k_{s_c\,powdery} \max(0, T_{0\,powdery} - T),$$

where $k_{s_c\,powdery}$ scales the temperature's influence, and $T_{0\,powdery}$ is the highest temperature influencing the critical angle. Then, the remaining proportion of snow $(1 - x_P)$ is split between unstable and stable snow, where the proportion of unstable snow is: $x_U = k_{unstable}(s - s_U)$, clamped between zero and one.

5.2. Snow state changes

Temperature [MGG*10] and slope [LLRR08] are prime determinants of change in snow stability and hence shifts between layer categories. We propose a simplified model that takes both into account.

Temperature induces melting of the snow cover according to:

$$D(t + \delta t_{\mathrm{m}}) = D(t) - \delta t_{\mathrm{m}}\, k_{\mathrm{melt}} \max(0, T - T_{0\,melt}),$$

where $k_{\mathrm{melt}} = 0.01 m\,day^{-1}\,^{\circ}C^{-1}$ is a global constant melting rate, $T_{0\,melt} = 0^{\circ}C$ is the minimum temperature at which melting occurs, and $\delta t_{\mathrm{m}} = 0.5$ day is the state change time step. Melting occurs layer by layer, progressing from top (unstable) to bottom (compact). The resulting water is not taken into account in our model.

Transitions in snow stability depend on temperature. At warmer temperatures (T_{warm}) snow becomes very unstable. At cool temperatures around freezing (T_{cool}) oscillations between partial melting and refreezing act to cement snow stability. At cold temperatures (T_{freeze}) snow stabilizes only under the pressure of its own weight. By experimenting with our model, we have found that this dynamic is captured by linearly interpolating stability changes depending on temperature: we use parameter values of $-k_w$, k_c, and 0 for warm, cool and cold temperatures, respectively, where $-k_w$ and k_c are user defined constants that relate to the speed of state change. Slope is also taken into account: instability increases slower on gentle slopes, and conversely stability increases slower on steeper slopes.

5.3. Diffusion of powdery snow

Because of its consistency, powdery snow undergoes an almost constant diffusion process of localized shifts and spills. We apply this diffusion of the powdery layer (P) at a high update frequency. Diffusion is computed for each cell $\mathbf{p}$ using the slope of each of the 4-cell neighbors $\mathbf{p}_n$:

$$s_d = (A(\mathbf{p}) - A(\mathbf{p}_n))/\|\mathbf{p} - \mathbf{p}_n\|.$$

The proportion of powdery snow shifted in direction d is a function of slope s_d in that direction, a constant rest slope value (s_{d_0}), and a shift rate parameter (k_m):

$$m_d(\mathbf{p}) = \begin{cases} \delta t_{\mathrm{d}}\, k_m \max(0, s_d - s_{d_0}) & \text{if } s_d \geq 0 \\ \delta t_{\mathrm{d}}\, k_m \min(0, s_d + s_{d_0}) & \text{otherwise,} \end{cases}$$

where δt_{d} is the time-step of the diffusion events. We define snow incoming i or outgoing o from $\mathbf{p}$ as the sum of all negative or positive m_d, respectively. Each outgoing positive m_d is scaled in proportion to the powder layer P by $(i + P)/o$ if $o > i + P$ to prevent negative quantities of powdery snow. The powdery layer is then updated by shifting snow from or to neighboring cells with respect to m_d.

5.4. Wind transport

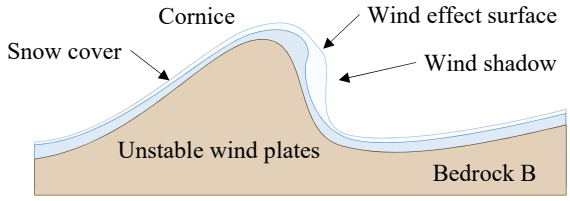

Figure 10: *The impact of wind on snow cover includes increased instability and formation of cornices.*

Wind acts on powdery snow to scour and advect it based on the curvature of the snow surface (Figure 10). To achieve this, we compute the wind resistance of the snow surface, taking into account wind velocity W and altitude A:

$$curv_W = |W_x|\frac{\partial^2 A}{\partial x^2} + |W_y|\frac{\partial^2 A}{\partial y^2}.$$

In practice, we zero $curv_W$ if it is positive, or if the wind-effect surface is distinct from the snow surface (as in the blue wind-shadow regions of Figure 8 right). Snow is then eroded from a cell in proportion to the concavity: the amount eroded is $\max(D, -k_{erosion}\, curv_W)$ and it is deposited to two of its neighbors in the direction of W (refer to Section 7 for implementation details).

Wind also weakens the stability of snow cover by tamping slopes, thereby forming a thin brittle shell over snow that would otherwise spill downslope. To account for this effect, we increase instability in proportion to positive changes in vertical wind speed, as illustrated in Figure 11.

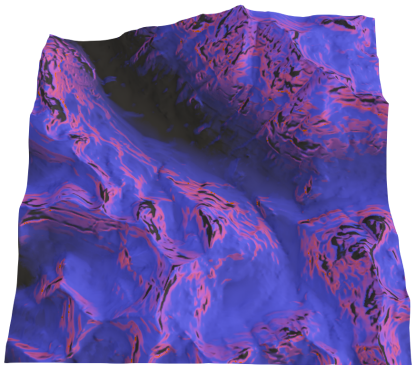

Figure 11: *Stability map: when present, unstable and powdery snow are shown in purple on top of compacted and stable snow in blue. Regions with no snow are in black.*

6. Interactive phenomena

The phenomena described in this section are either triggered automatically (with a low per-cell probability so that their occurrence is rare), or manually by the user selecting a seed point. The latter is most useful if real and simulation time are synchronized so that users can best judge the dynamics of the phenomenon.

6.1. Avalanches

Avalanche behavior involves many correlated conditions, including terrain shape and initial environmental conditions, and a variety of constituents, including entrained air and different forms of snow. Here, we focus on two principle avalanche types: (1) *Dry snow avalanches* composed of powdery snow mixed with cohesive ice-blocks. Such avalanches are best treated as a flow of granular material; (2) *Wet snow avalanches* with heavy, part-melted snow that behaves as a viscous fluid. These types represent two poles of a continuum: our unified treatment allows a mix of behaviors parameterized by temperature.

We should note that snow type (either wet or dry) is only one axis in the standard classification of snow avalanches [MS06]. Other axes of the zone of origin include the type of start zone (point or slab) and the level of the sliding layer (surface or full). Within this classification scheme we support surface sliding point zone avalanches for both wet and dry snow types.

Fluid simulation. Both avalanche types can be encompassed by a fluid simulation and for this we choose to implement a hydrostatic pipe-model from [OH95,ŠBBK08]. This 2D Eulerian method discretizes the simulated domain into 2D cells corresponding to columns of snow that are connected by virtual pipes. A pipe's pressure depends on the fluid content, and the simulation stabilizes the flow of fluid induced by pressure differences. For a given cell C in the terrain grid, we use Moore's neighborhood, which considers all eight neighboring cells. The difference in pressure ΔP_i between two neighboring cells C and C_i is then

$$\Delta P_i = \rho\, g\, (A(C) - A(C_i)),$$

where $A = B + D$ is the altitude of the uppermost surface, ρ is the density of the fluid, and $g = 9.81$ is the gravitational constant. The acceleration a_i of the snow between neighboring cells is

$$a_i = \Delta P_i / (\rho \mathrm{d}x),$$

where dx is the cell width. The flow in the pipe evolves as

$$\phi_{C\to C_i}(t+\delta t) = \phi_{C\to C_i}(t) + \delta t\, c\, a_i,$$

with c being the cross section of the pipe (set as constant as $c = dx^2$)

Finally the height of the unstable snow layer U is adjusted according to:

$$U(t+\delta t) = U(t) - \delta t\, dx^{-2} \sum_i \phi_{C\to C_i}(t). \tag{3}$$

To enforce snow removal by a positive amount, we use a rescaled version of $\phi_{\mathbf{p}\to\mathbf{p}_i}(t+\delta t)$ in Eqn (3), with the scaling factor: $\min(V_{out}, dx^2 U(t))/V_{out}$, where V_{out} is the total outflow and:

$$V_{out} = \delta t \sum_i \max(0, \phi_{C\to C_i}(t+\delta t)).$$

Granular material flows are commonly modeled by adding a yield criterion to the fluid, i.e. a friction force opposing the flow, of norm:

$$|\mathbf{f}| = g \tan(\theta_c),$$

and where θ_c is the rest angle of the snow. If the flow is null, or applying this force would reverse flow direction, then flow is set to zero to model static friction. Such a yield criterion can introduce gridding artifacts in an Eulerian simulation. However, in our case the avalanche footprint is small relative to the terrain and no visual artifacts are apparent. The viscosity of the avalanche is obtained by adding a viscous force, opposed to the flow direction $\phi_{\mathbf{p}\to\mathbf{p}_i}$:

$$\mathbf{v} = -k_{\mathbf{v}}\, U\, \phi_{\mathbf{p}\to\mathbf{p}_i},$$

We augment the method of Šťava *et al.* [ŠBBK08] by adding these forces in the acceleration term:

$$a_i = \Delta P_i/(\rho \mathrm{d}x) + \alpha \mathbf{f} + (1-\alpha)\mathbf{v}$$

where $\alpha \in [0,1]$ is proportional to snow temperature.

Application to our framework: In our simulations, only the unstable layer U and the powdery snow P on top (if not yet diffused) are subject to avalanche redistribution. It is assumed that the stable and compacted layers are too strongly fused (making our avalanches of surface layer avalanche type).

In our framework, we use a *Snow Moving* boolean layer to decide if unstable snow is locally still or in motion. The snow is initially still. A *rupture point* can then be triggered either automatically or by the user. If the unstable layer U is non-empty, then the avalanche rapidly propagates to neighboring areas, with reach proportional to the thickness of U. All unstable snow in the rupture zone is immediately set into motion, as well as all unstable snow in the downslope reach of the avalanche.

6.2. Ski tracks

Human action on a terrain is another non negligible factor in modeling realistic mountainous landscapes. In particular, Nordic skiing not only compresses snow layers and leaves tracks, which is important for a more authentic feel to the scene, but can also trigger avalanches when performed on unstable layers of snow.

The size of our simulation grid (10 m per cell) only allows a consideration of the general direction of the skiers. While this is sufficient when modeling the impact of Nordic skiing on snow state during the simulation, we need more precise paths to achieve realistic rendering. We thus opted to integrate a procedural method to generate plausible refined tracks.

Global path search. As with avalanches, skiers that are not user-triggered are automatically generated by a Ski-Track-Start event called on the terrain, where each cell has a low probability of spawning a skier. This probability is influenced by the length and viability of a ski route. Therefore, we pre-compute a map registering the distance from each cell to its farthest down-slope end point (local minima or edge of the terrain). We use a simple cellular automaton that finds minima on the terrain and propagates the distance to them in subsequent steps to efficiently compute this map, directly on the GPU. This distance map is used both as a weighted

mask when automatically spawning new skiers, and as a guide to discourage skiers from reaching dead-ends, as discussed next.

Large scale paths are computed to approximate general direction and movement of the skiers, without the detail of the curves used to regulate their speed. For that, we take into account the slope of the mountain by defining an ideal slope angle s_t that skiers will be comfortable with and try to follow, and make sure that there is enough snow for them to ski on. In this work, skiers are modeled as independent agents, defined by their position and orientation, responsible for deciding their next move using a local search, based on the amount of snow present in neighboring cells, the corresponding relative slope s_n and a pre-computed weight w_d related to the distance to a terminus. In practice, skiers have a small lookahead window of a few tiles, and will steer in the best candidate direction defined by the center of a nearby cell. The probability for a skier aiming towards a given cell n is:

$$P(n) = \mathbf{1}_{snow>threshold}\, f(|s_n - s_t|)\, w_d, \tag{4}$$

where $\mathbf{1}$ is the indicator function, s_n is the slope between the current cell and cell n, and f is a function that assigns a weight, which can be changed to tune the behavior of skiers with regard to slope. To avoid sharp changes in direction, a smooth transition to the new steering direction computed and applied to the orientation of the skier. The steering direction is re-evaluated at each animation step.

Refined tracks are created for rendering purposes. They are approximated based on the observation that the precise movement of skiers is analogous to sine waves of varying amplitude and frequency, with sharp turns used to slow down and straight paths to gain speed. With this in mind, we model movement on each straight segment of a global ski trajectory using $\mathbf{p}(t) = a\sin(2\pi f t)$, where a is the amplitude and f is the frequency, dynamically updated with the terrain's varying slope.

Indeed, a skier moving straight down a mountain will go faster than one with a trajectory following the isoline. To account for this, we compute the effective slope s_e of the skier's trajectory as:

$$s_e = \arcsin \frac{\sin(s_n)}{2fl} \qquad l = \sqrt{\frac{1}{4f^2} + 4a^2},$$

where l is the distance between sine curve extrema. This provides the local frequency value required for skiers to reach their comfortable target slope s_t:

$$f = \frac{\sqrt{\sin^2(s_n) - \sin^2(s_t)}}{4a\sin(s_t)}$$

Continuity with the previous refined position and orientation of the skier is ensured by choosing an appropriate starting phase value along the sine curve. At each animation step, the resulting movement detail is mapped on the fly onto the lower resolution trajectory computed using Eqn. (4). This is done using a local update to a ski-tracks texture layer covering the whole terrain. An alternative is to export this texture as a displacement map for off-line rendering.

Interaction between ski tracks and snow is two-way. Once a skier enters a cell it transforms a fixed amount of snow from unstable to stable, or from stable to compacted if no more unstable snow remains. If there is unstable snow still remaining then the probability of an avalanche is increased. Conversely, the impact of snow on rendered ski tracks is taken into consideration: as snow is deposited along the path or shifted by the wind, the tracks fade dynamically depending on the quantities involved.

7. Implementation

Our system is implemented in C++ with use of OpenGL and GLSL for rendering and CUDA for simulation (source code is provided.) All examples were measured on a laptop equipped with an NVidia Quadro K2100M GPU. Offline rendering was done with Terragen software.

7.1. Wind transport and diffusion

Wind transport and diffusion of powdery snow can move snow from the current cell to one of its neighbors. This can cause race condition if the simulation executes on all cells concurrently. To avoid this situation, we run the kernel in multiple passes, each time affecting only cells that would not cause parallel write conflicts. We perform the simulation on cells according to a cross-shaped tiling, which requires five kernel launches (one per arm and one at the cross center). This is a very tight arrangement of kernels that allows for conflict free usage of direct neighboring cells. In our experiments, this implementation gave slightly better performances than using atomic instructions, and was also both easier to implement and more readable.

5	1	2	3	4	5
2	3	4	5	1	2
4	5	1	2	3	4
1	2	3	4	5	1
3	4	5	1	2	3
5	1	2	3	4	5

7.2. Avalanches

Our GPU implementation is inspired by the work [ŠBBK08], with the distinction that we store four floating points values for signed flow instead of the eight outflows: our GPU implementation allows writing to neighboring cells, as long as two cells are not written simultaneously. When computing the flow for the given cell **p**, the resulting flows are written at **p** only if they are positive, and in the neighboring cells if they are negative. In this way, all the flows are updated and race conditions are avoided while maintaining high occupancy.

7.3. Ski tracks

Ski tracks are also computed on the GPU. Skiers are spawned in parallel in each cell with a small probability, and then moved still in parallel on a cell-by-cell basis using the same tiling pattern described for wind transport. For large-scale paths, we only store the number, position an orientation of skiers present in each cell, and every thread dispatches its skiers to neighboring cells. For fine-scale tracks, we also store the frequency and phase of the sine curve used to refine each straight segment, since this is necessary to render continuous curves. An alpha-blending coefficient, updated when snow moves into a cell, is used to progressively fade-out ski tracks.

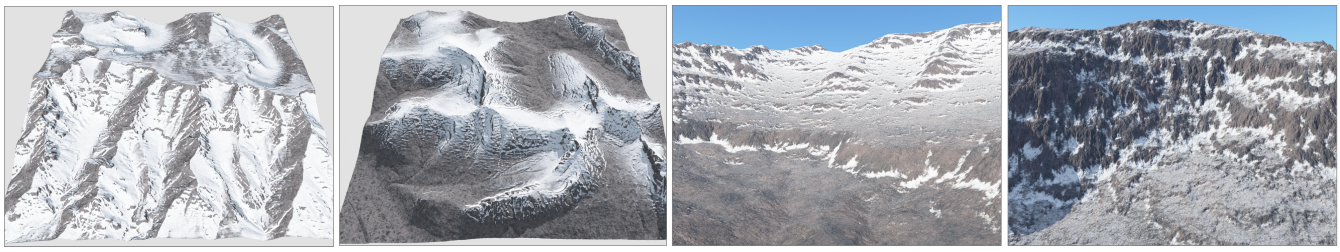

Figure 12: *Various results, highlighting the effects of sun exposure, diffusion and slope.*

7.4. Performance

Temperature and wind calculations are the most computationally demanding. They are performed once in entirety as a preprocessing step and require approximatively 7.8 and 10.5 seconds, respectively, to compute. The simulation and the vizualization are done on the GPU. We achieve interactive simulation rates of 30Hz, on a 1024×1024 grid (Table 1). Wind transport and powdery snow diffusion are relatively slow because they requires five kernel launches each. In contrast, precipitation and snow-state events are less demanding since they only require a single kernel launch.

Event	Time	n	Tot. time	%
Wind transport	60	2	120	10
Powdery snow diffusion	64	2	128	11
Avalanche	38	10	380	32
Snowing	12	1	12	1
Snow state change	16	2	32	3
Skier movement	50	10	500	43
Total	240	37	1 172	100

Table 1: *Average time for one frame (in ms), number of occurrences of events n, total time (in ms), and percentage of the total for each stage on a* $1\,024 \times 1\,024$ *grid. In practice, the snow cover evolution runs for about* 100 *frames (days). An accumulated* 1 000 *shorter frames (representing fractions of a second) are used to visualize avalanche and ski tracks. Subsets of these frames are packed along the duration of such shorter-term events, usually between two longer-term frames.*

Because we want to display a regular temporal evolution of one frame per day (except during temporal zoom), the number of occurrences of each event per day impacts the performance. In our implementation, we perform on average 200 wind transport, powdery snow diffusion and melting events, 100 precipitation events, and 300 avalanche events over 100 days. We achieve an interactive time step of half a second, and the user has to wait about 60 seconds for the final landscape. When the user manually launches avalanches or ski tracks, the temporal zoom suppresses occurrence of all the other events, so that the interactive phenomenon is displayed in real time.

8. Results and discussion

We have sourced mountainous DEM terrains from the U.S. Geological Survey, specifically from on or near: Steens Mountain, Oregon; Glacier Park, Montana; the Teton Range, Wyoming; and Mount Elbert, Colorado. Unless otherwise indicated, the terrain and simulations are sampled at 10 m cell resolution and have an extent of 10×10 km. In addition, we compare against real snow depth data provided by the NASA Airborne Snow Observatory for Tuolumne Basin, California and Conojes Basin, Colorado at 10 m sampling, and Bassies in the Pyrenees, France [MGB*16] at 2 m sampling.

8.1. Visual results

Here we showcase a variety of outcomes that are visually consistent with underlying mountainous terrain. Figure 12 shows both far and near landscape renderings of (from left to right): a high-altitude region near Steens Mtn., with sun-exposed slopes; a sparser covering of an area from Glacier Park; a higher-resolution simulation of Steens Mtn., highlighting wind impact; and another subsampling showing snow collected below cliffs. The latter two simulations are at 2 m resolution over $2 \times 2\,km$. Collectively, these results demonstrate snow deposition and melting due to snowfalls and state changes, and how diffusion and avalanches clear steep slopes.

Figure 13 shows summer, autumn, late winter, and spring melt stages for a landscape subject to an evolving user-specified weather scenario. In autumn, snow settles at high altitudes on gentler sun-shadowed slopes. In the heart of winter, even steep slopes are covered and cornices develop. In spring, lower, flatter areas melt first, with snow retained at high altitudes and for larger deposits.

8.2. Validation - perceptual user study

For validation purposes, the ideal would have been to compare our simulated snow coverage results against real snow coverage data on the same terrain topography. Unfortunately, to achieve such a match and set our simulation parameters would require significant additional information on terrain type, water bodies, moisture levels, ice coverage, prevailing winds, and human activity. Instead, we validated physical plausibility through a perceptual user study. We chose to compare results against both real snow data and a previous method, using a two-alternative forced choice (2-AFC) user experiment. For comparison to prior work, we selected the occlusion method of Foldes and Benes [FB07] since this method is intended for landscapes of a similar scale and requires comparable computation. Moreover, since we had no physical ski-track data or previous

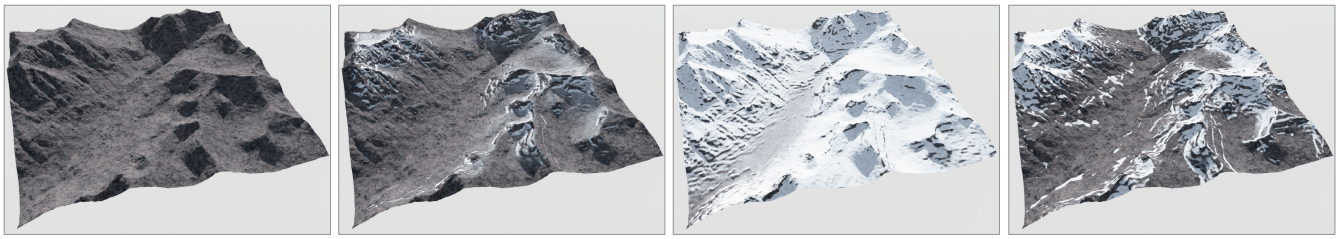

Figure 13: *Dynamic landscape, with from left to right: bare rock in summer, the first autumn snows, snow in late winter, and melting in spring.*

Figure 14: *Specific effects simulated by our method.*

ski track models to compare against, we excluded ski-tracks from the user study.

Participants in the study were presented with nine pairs of images, and asked to select the most *natural* in each case. All images where rendered from synthetic or real data using the same render settings, with image pairs drawn from the set of {real data, our results, ambient occlusion} such that each two-element subset appeared three times. Care was also taken to use the same mountain range an equal number of times, with the data sources as indicated previously. The overall order of presentation, as well as the order within pairs, was randomized (some examples are show in Figure 15).

We had 57 participants between the ages of 21 and 78. Some had significant computer graphics exposure, or were accustomed to high altitude winter scenery. Protocol, data and results are available as the additional material of the paper. Our experiment shows that overall a particular generation method was selected on average (with standard deviation σ), as follows: real data 68.2% ($\sigma = 17\%$), our results 47.8% ($\sigma = 26\%$), and previous ambient occlusion 33.9% ($\sigma = 22\%$) of the time. This indicates that, although we improve markedly over previous work (in particular participants favored our results over previous work 67% of the time on average), our simulation is not yet indistinguishable from physical reality. This is likely due to certain secondary effects that we do not consider, such as trees, soil constituency, running water, and surface ice formation.

With a more extensive user study it would be possible to evaluate the visual impact of the individual components and parameters of our model. Although time consuming, such a study would be a useful extension for future work.

9. Conclusion

We have presented the first modeling method for dynamic, snow-covered landscapes. Thanks to our stochastic, Poisson-based simulation of diverse phenomena, we capture complex physical interactions, such as snow melted by both direct and indirect sunlight, powdery snow shifted by the wind, and avalanches initiated in steep areas of unstable snow. All these events combine to sculpt the landscape over time, leading to the formation of visually consistent features, such as overhangs of snow under ridges, piles of snow at the base of slopes cleared by avalanches, or ski-tracks consistently snaking down-slope. Our method allows for interactive modeling that supports real-time phenomena. We have validated our method with a user study and have shown a variety of practical examples.

We have focused primarily on bedrock and snow and have thus neglected many secondary effects. In particular, trees, rocks, soft soil, grass, ice formation and running water are important for snow formation and interaction and should be addressed as future work. Moreover, modeling slab snow - a cohesive layer of snow sitting on a weaker layer - and more varied tracks would have been a possibility. Incorporating such effects, at whatever individual simulation scale is required, should be made easier by our Poisson-based stochastic simulation framework.

Lastly, our use of different landscape scales, with cell widths from 2 to 10 meters, demonstrates that the method is robust to scale changes between simulation scenarios. However, an interesting future avenue would be to incorporate a pyramidal approach capable of handling both large scales and fine details within the same terrain.

Acknowledgements

We thank the NASA Airborne Snow Observatory (http://aso.jpl.nasa.gov) and Simon Gascoin for topographic and snow

Figure 15: *Validation by forced-choice comparison between real data, our results and an existing ambient occlusion-based techniques. Each row represents a typical pairing with randomized order as presented in our user study.*

data. This material is based upon work supported by the National Science Foundation under Grant No. (#1747544).

References

[CGG*17] Cordonnier G., Galin E., Gain J., Benes B., Guérin E., Peytavie A., Cani M.-P.: Authoring landscapes by combining ecosystem and terrain erosion simulation. *ACM Trans. Graph. 36*, 4 (2017). 3, 5

[CSLW03] Chen Y., Sun H., Lin H., Wu E.: Modelling and rendering of snowy natural scenery using multi-mapping techniques. *Journal of Visualization and Computer Animation 14*, 1 (2003), 21–30. 3

[DGP16] Dagenais F., Gagnon J., Paquette E.: An efficient layered simulation workflow for snow imprints. *The Visual Computer 32*, 6-8 (2016), 881–890. 2

[FB07] Foldes D., Benes B.: Occlusion-based snow accumulation simulation. In *Vriphys'07* (2007), pp. 35–41. 1, 3, 5, 10

[Fea00] Fearing P.: Computer modelling of fallen snow. In *Proceedings of Siggraph* (2000), pp. 37–46. 1, 2

[FG09] Festenberg N. V., Gumhold S.: A geometric algorithm for snow distribution in virtual scenes. In *Proceedings of EG Conference on Natural Phenomena* (2009), pp. 17–25. 3

[FG11] Festenberg N., Gumhold S.: Diffusion-based snow cover generation. *Comp. Graph. Forum 30*, 6 (2011), 1837âĂŞ–1849. 3

[FO02] Feldman B. E., O'Brien J. F.: Modeling the accumulation of wind-driven snow. In *ACM SIGGRAPH 2002 Conference Abstracts and Applications* (2002), SIGGRAPH '02, pp. 218–218. 2

[GPG*16] Grosbellet F., Peytavie A., Guérin E., Galin E., Mérillou S., Benes B.: Environmental objects for authoring procedural scenes. *Comp. Graph. Forum 35*, 1 (Feb. 2016), 296–308. 3

[HM09] Hinks T., Museth K.: Wind-driven snow buildup using a level set approach. In *EG Ireland Workshop Series* (2009), pp. 19–26. 2

[KL03] Kim T., Lin M. C.: Visual simulation of ice crystal growth. In *Proceedings of the Symp. on Computer Animation* (2003), SCA '03, pp. 86–97. 1

[LLRR08] Lehning M., Löwe H., Ryser M., Raderschall N.: Inhomogeneous precipitation distribution and snow transport in steep terrain. *Water Resources Research 44*, 7 (2008). 7

[MaMAL05] Moeslund C., anvisud Madsen T., Aagaard M., Lerche D.: Modeling falling and accumulating snow. In *Vision, Video and Graphics* (2005). 2

[MC95] Masselot A., Chopard B.: Cellular automata modeling of snow transport by wind. In *International Workshop on Applied Parallel Computing* (1995), pp. 429–435. 2

[MC00] Muraoka K., Chiba N.: Visual simulation of snowfall, snow cover and snowmelt. In *Proceedings of the Parallel and Distributed Systems: Workshops* (2000), ICPADS '00. 2

[MGB*16] Marti R., Gascoin S., Berthier E., de Pinel M., Houet T., Laffly D.: Mapping snow depth in open alpine terrain from stereo satellite imagery. *The Cryosphere 10*, 4 (2016), 1361–1380. 10

[MGG*10] Maréchal N., Guérin E., Galin E., Mérillou S., Mérillou N.: Heat transfer simulation for modeling realistic winter sceneries. *Comp. Graph. Forum 29*, 2 (2010). 2, 5, 7

[MS06] McClung D., Schaerer P.: *The Avalanche Handbook*. The Mountaineers, 2006. 8

[MT10] Moriya T., Takahashi T.: A real time computer model for wind-driven fallen snow. In *ACM SIGGRAPH ASIA 2010 Sketches* (2010), pp. 26:1–26:2. 3

[NIDN97] Nishita T., Iwasaki H., Dobashi Y., Nakamae E.: A Modeling and Rendering Method for Snow by Using Metaballs. *Comp. Graph. Forum* (1997). 2

[OH95] O'brien J. F., Hodgins J. K.: Dynamic simulation of splashing fluids. In *Computer Animation'95* (1995), IEEE, pp. 198–205. 8

[PTS99] Premože S., Thompson W. B., Shirley P.: Geospecific rendering of alpine terrain. In *Eurographics Conference on Rendering* (1999), EGWR'99, pp. 107–118. 3

[RGS09] Ritschel T., Grosch T., Seidel H.-P.: Approximating dynamic global illumination in image space. In *Proceedings of the Interactive 3D graphics and games* (2009), ACM, pp. 75–82. 5

Constant	Description	Value
T_0	Sea level temperature	-
k_t	Temp. offset with altitude	$-0.01°C\,m^{-1}$
k_i	Temp. offset per daylight hour	$0.41°C\,h^{-1}$
k_{sun}	Amount of direct illumination	$0.9\,h$
k_{sky}	Amount of ambient illumination	$0.1\,h$
k_{ind}	Amount of indirect illumination	$0.9\,h$
$k_{venturi}$	Wind speed offset with altitude	$10^{-3}\,m^{-1}$
$k_{terrain}$	Topography effect on wind	0.5
$\|\mathbf{W}\|$	Wind strength at sea level	$10\,m\,s^{-1}$
c	Decrease of wind effect vs hight	$0.7\,m\,s^{-2}$
k_{snow}	Snow fall strength	$10^{-4}\,day^{-1}$
$A_{0\,precip}$	Rain-snow limit altitude	-
$k_{powdery}$	Powdery snow from precipitation	5
s_{c_0}	Powdery snow min critical slope	0.5
$k_{s_c\,powdery}$	Temp. influence on s_{c_0}	$5\times10^{-2}\,°C^{-1}$
$T_{0\,powdery}$	Highest temp. that affects s_{c_0}	$-10°C$
$k_{unstable}$	Unstable snow from precipitation	1
s_U	Critical slope of unstable snow	0.3
k_{melt}	Melting rate	$0.01m\,day^{-1}\,°C^{-1}$
$T_{0\,melt}$	Melting temp.	$0°C$
T_{warm}	Temp. of the unstable snow	$5°C$
T_{cool}	Optimal temp. for stable snow	$-5°C$
T_{freeze}	Min temp. affecting stability	$-20°C$
k_w	Instability induced by warmth	$10^4\,m\,day^{-1}$
k_c	Stability induced by cold	$10^4\,m\,day^{-1}$
s_{d_0}	Rest slope for snow diffusion	0.5
k_m	Diffusion rate	$0.5day^{-1}$
$k_{erosion}$	Wind erosion rate of snow	$0.1m\,s^{-1}\,day^{-1}$
ρ	Snow density	$0.5g\,cm^{-3}$
g	Gravitational acceleration	$9.8m\,s^{-2}$
θ_c	Rest angle of avalanche snow	30°
$k_{viscous}$	Snow viscosity	$5\times10^{-4}\,s^{-2}$
s_t	Target slope for skiers	30°
a	Tracks sine wave target amplitude	$5\,m$
f	Tracks sine wave target frequency	$0.01\,m^{-1}$

Table 2: *Simulation parameters with symbol, meaning, and default value (including physical units, "Temp." stands for temperature). Sea level temperature T_0 and rain-snow limit altitude $A_{0\,precip}$ are highly variable depending on the input terrain and are set by the user.*

[RLD15] Reynolds D. T., Laycock S. D., Day A. M.: Real-time accumulation of occlusion-based snow. *The Visual Computer 31*, 5 (2015), 689–700. 3

[ŠBBK08] Šťava O., Benes B., Brisbin M., Křivánek J.: Interactive terrain modeling using hydraulic erosion. In *Proceedings of the Symp. on Computer Animation* (2008), pp. 201–210. 8, 9

[SEN07] Saltvik I., Elster A. C., Nagel H. R.: Parallel methods for real-time visualization of snow. In *Proceedings of the PARA* (2007), pp. 218–227. 2

[SKIT15] Sai-Keung W., I-Ting F.: Hybrid-based snow simulation and snow rendering with shell textures. *Computer Animation and Virtual Worlds 26*, 3-4 (2015), 413–421. 2

[SOH99] Sumner R. W., O'Brien J. F., Hodgins J. K.: Animating sand, mud, and snow. *Comp. Graph. Forum 18*, 1 (1999), 17–26. 3

[SSC*13] Stomakhin A., Schroeder C., Chai L., Teran J., Selle A.: A material point method for snow simulation. *ACM Trans. Graph. 32*, 4 (2013), 102:1–102:10. 2

[Tok06] Tokoi K.: A shadow buffer technique for simulating snow-covered shapes. In *International Conference on Computer Graphics, Imaging and Visualisation* (2006), pp. 310âĂŞ–316. 3

[WWXP06] Wang C., Wang Z., Xia T., Peng Q.: Real-time snowing simulation. *Vis. Comput. 22*, 5 (May 2006), 315–323. 2