CGI2016 manuscript No.
(will be inserted by the editor)

An Efficient Layered Simulation Workflow For Snow Imprints

François Dagenais · Jonathan Gagnon · Eric Paquette

Abstract This paper introduces a novel workflow to generate snow imprints, and model the interaction of snow with dynamic objects. We decoupled snow simulation into three components: a base layer, snow particles, and snow mist. The base layer consists of snow that has not been in contact with a dynamic object yet, and is stored as a level set. Snow particles model the interaction between the snow and the dynamic objects. They are added when the dynamic objects collide with the base layer, and are animated using an adapted granular material simulation. The very thin and powdery snow released by airborne snow particles is modeled by the snow mist. This component is greatly influenced by the surrounding air medium, thus it is animated using a fluid simulation. This decomposition allows to focus memory expensive and time-consuming computations only where dynamic objects interact with the snow, which is much more efficient than relying on a full-scale simulation.

Keywords animation, fluid simulation, natural phenomena, snow, imprints, footprints

1 Introduction

Snow is an important aspect that is ubiquitous in movies with winter scenery. While modeling static snow can be an easy task, simulating its interaction with characters and dynamic objects requires much more time and efforts in order to achieve a visually realistic look.

Simulators available in current commercial software and state of the art simulation methods for snow and granular material [8, 10, 16, 19, 25] can simulate snow at a limited scale. They require a large amount of simulation data to generate realistic results even just at a moderate scale, which results in a large memory consumption and long computation times. Furthermore, in larger scenes, a considerable amount of the memory and computation will most likely involve snow that remains unchanged throughout the simulation. Typical scenarios, like character footsteps or sleigh tracks, interact with a small subset of snow. In such cases where the interactions with the snow are localized, height map methods [17, 18, 24] limit their computations to areas affected by the motion of dynamic objects. Relying on a much more efficient simulation strategy, these methods significantly reduce the memory and computation costs. However, they are limited to a simple deformation of a flat surface, thus they cannot model snow that is projected in the air.

This paper introduces an efficient workflow to handle the interaction between snow and dynamic objects by subdividing the phenomenon into three separate simulations (see Fig. 1). First, our base layer simulation uses constructive solid geometry (CSG) operations on level set representations of the snow and the dynamic objects to leave smooth imprints and determine locations where finer resolution simulations are locally required. Then, a modified granular simulation adds snow particles to model the behavior of snow that gets pushed and tossed. Finally, a tailored fluid simulation replicates the look and the behavior of the airborne

F. Dagenais[1]
francois.dagenais.2@ens.etsmtl.ca

J. Gagnon[1,2]
jonathangagnon@gmail.com

E. Paquette[1]
eric.paquette@etsmtl.ca

[1]Multimedia Lab, École de technologie supérieure, Montreal, Canada
[2]Mokko Studio, Montreal, Canada

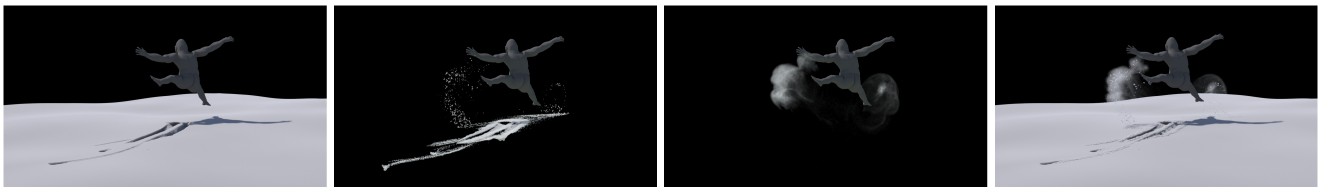

Base layer Snow particles Snow mist Final result

Fig. 1 Our decomposition into three components allows us to model efficiently the fine scale details of snow.

sparse snow mist driven by the surrounding air medium. The contributions of this paper are:

– A decoupling of the behavior of the snow into three components;
– An efficient simulation approach for each of the three components;
– Adapted simulations taking into account the impact of the air resistance for the snow particles and mist;
– A control over the transfer of snow from the base layer to the snow particles and then to the snow mist.

Given our cascade of simulation components, our approach provides early feedback of the look of the imprints and the movement of the snow particles. It handles the snow interaction with simple or complex dynamic objects, it can cope with large scenes, and it provides good visual results. Furthermore, it significantly decreases the memory consumption and the computation times by adding details only where they are required. Finally, it provides separate control over three different components of the snow behavior.

2 Related Work

Early work on snow focused mostly on modeling its accumulation on the ground and objects. Some methods analyzed the flow of snow [4,12] as it falls from the sky, while others relied on geometric models [6,7] to generate the snow in a static scene. Some work even used physical simulations to model the impact of wind [5,22] and heat transfer [11]. Nevertheless, these methods focus mainly on the generation of a static geometry that can grow over time, as snow accumulates. Thus, they do not address the animation of snow that interacts with dynamic objects.

It is quite common to consider snow as a granular material, like sand. Sumner et al. [20] used a height map to animate sand, mud, as well as snow. In their method, the snow height map is updated when a collision occurs with a dynamic object. Cells surrounding the colliding cells are then updated to take into account mass displacement and erosion. Particles are created at the surface of the colliding objects to model snow that adheres to these objects and then falls. As particles fall back to the ground, volume is added to the corresponding height map cells. This method was improved by Onoue and Nishita [17], by allowing snow to lie on top of objects through the use of a multivalued height map. Zeng et al. [24] further improved the behavior by considering the velocity of the dynamic objects when computing the mass displacement. Pla-Castells et al. [18] used cellular automata to compute the mass displacement between height map cells in a more physically correct way. These approaches were designed for real-time applications and are fast to compute, however, they are very limited in the variety of behaviors they can reproduce. The height map behavior is limited to a 2D simulation, and cannot model the dynamics of airborne snow. While some of these methods [17,20] use particles to depict airborne sand or snow, their dynamics is very limited. Inter-particle interactions are not considered, and particles do not interact with dynamic objects after being emitted. Furthermore, the transfer of mass from particles to height map cells will likely produce temporal artifacts in the animation.

Fully physically-based simulations handling the interaction between particles and dynamic objects enable the complete range of snow dynamics. Bell et al. [2] used the discrete element method (DEM) to simulate sand using particles. The long computation times of their method were shortened significantly in the work of Zhu and Yang [25] by converting still particles hidden under surface particles to a multivalued height map, and only simulating the surface particles. Fluid simulation approaches have also successfully been adapted to simulate granular materials. A variety of paradigms have been used, such as Fluid Implicit Particle (FLIP) [16, 26] and Smoothed Particle Hydrodynamics (SPH) [1,8, 9]. Additionally, position based dynamics [13] have been used by Macklin et al. [10] to simulate sand. While these simulation paradigms enable a rich range of dynamic behaviors, they do not model some of the characteristics specific to snow, such as its compressibility and its interaction with air.

Instead of simulating any granular material, some fully physically-based simulation methods specifically address the simulation of snow. Takahashi and Fujishiro [21] extended the SPH method to handle the sintering of snow. Stomakhin et al. [19] used the Material Point Method (MPM) to simulate wet and dense snow. Their hybrid Eulerian/Lagrangian approach allowed them to achieve a visually plausible simulation for snow that exhibits both fluid and solid properties. Wong et al. [23] simulated snow using a DEM simulation with cohesive forces. Similarily to Zhu and Yang [25], they reduce computation times by only considering surface particles. While these simulation methods can model a wide range of snow behaviors, their computation time and memory requirements become unmanageable when considering larger scenes or when requiring finer details. Thus, they generally scale poorly in scenes that cover a large area. On the other hand, as stated previously, methods based on multivalued height maps are limited in the range of behaviors that they can model, but are much faster and scale better to larger scenes.

As multivalued height map methods, our approach has low computation time and memory costs. Nevertheless, we rely on a level set instead of a grid, resulting in more flexibility and precision for the dynamic object interactions. The level set also allows us to achieve a higher degree of realism by applying a 3D localised noise where imprints appear in the snow. Additionally, to allow for the wider range of behaviors of the fully physically-based methods, our approach proposes an adapted and localized granular simulation. Furthermore, the computation times of our simulation are lowered by computing its dynamics only where potential interactions may occur. Finally, to reproduce the full spectrum of snow behavior, our snow mist uses a modified fluid simulation modeling the interaction with the surrounding air. Thus, our approach allows more visually complex results than multivalued height maps methods, while requiring less memory and computation times than a full particle simulation.

3 Overview

Our approach aims to simulate powdery snow, which gets compressed under the weight of objects, such as a character's feet, and also exhibits complex behaviors when it interacts with objects or becomes airborne. As shown in Fig. 2, to achieve such a behavior we decompose the physical process into three components: the base layer (see Sec. 4), the snow particles (see Sec. 5), and airborne snow mist (see Sec. 6). We use separate simulation paradigms for each of the components; When brought together, these three components provide a very efficient and easy to control snow imprint workflow.

The first component, which we call the base layer, is made of the snow that did not interact with the dynamic objects yet. Handling it is efficient, as it is processed only when the objects collide with it. In such event, the snow of the base layer is compressed by the dynamic objects, and some of that snow is transferred to the second component: the snow particles. Those moving snow particles reproduce the complex behavior of snow that interacts with the dynamic objects, other snow particles, and the base layer. Driven by a particle simulation, they enable the accumulation of snow as well as the projection of snow in the air. While these snow particles mimic the behavior of snow clumps of modest size, they fail to recreate the motion of tiny airborne snow elements, whose behavior is strongly affected by the supporting air medium. Hence, the airborne snow particles trigger the creation of the third component: the snow mist. This last component relies on an incompressible fluid simulation, and its behavior relies on that of the surrounding air medium.

To maintain a realistic and efficient simulation, we rely on a unidirectional coupling between the three components; the base layer influences the snow particles, which in turn influence the snow mist. This brings the advantage of providing early feedback: the base layer simulation can be entirely computed before simulating the snow particles, which can in turn be completely fine tuned before generating the snow mist.

4 Base Layer

The initial snow layer can be designed in two ways: by specifying a height on top of the objects and ground, or by specifying the top surface of the snow. The base layer is constructed from this designed snow shape. Then, throughout the animation, the parts that collide with the dynamic objects are removed to match the contour of the volume swept by the objects. This volume loss replicates the compression of the snow under external forces. To have a fast computation, we found that applying CSG operations on a level set [14] provides the versatility needed to model realistic imprints of the dynamic objects. This level set is updated, once per frame, by compacting the regions swept by the dynamic objects. In order to do so, a second level set is built using the dynamic objects, and is then subtracted from the base layer. If no special treatment is done, "dents" will be visible because of the discrete positions in time. This problem can be seen in Fig. 3(a), where a box is quickly moved through the base layer of snow.

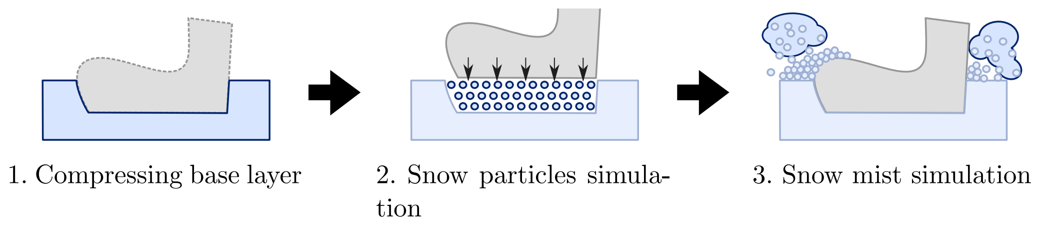


Fig. 2 Our approach consists of three different simulations, each related to a different component of our physically-based process decomposition.

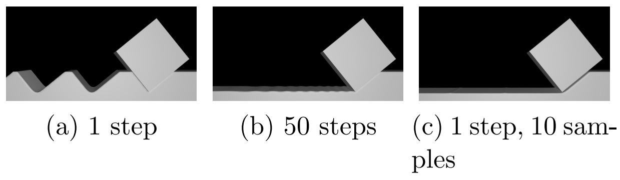


Fig. 3 A moving box sweeping through the base layer. (a) With only 1 step per frame, the full motion of the object is not well captured. (b) Even with a large number of simulation steps, small "dents" can be seen. (c) Our approach provides smoother results with only 1 step and a few samples.

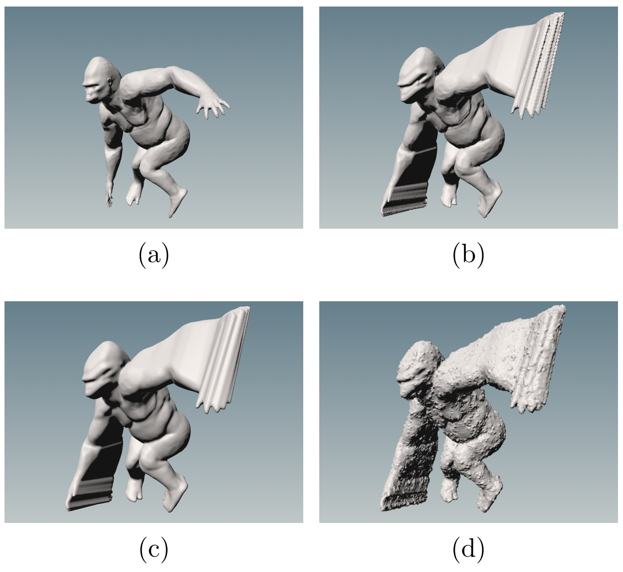


Fig. 4 Before being subtracted from the base layer, dynamic objects undergo several operations. The original geometry (a) is sampled multiple times throughout its motion (b). It is then voxelized, before being dilated and smoothed (c). Finally, a noise displacement is applied on its surface (d).

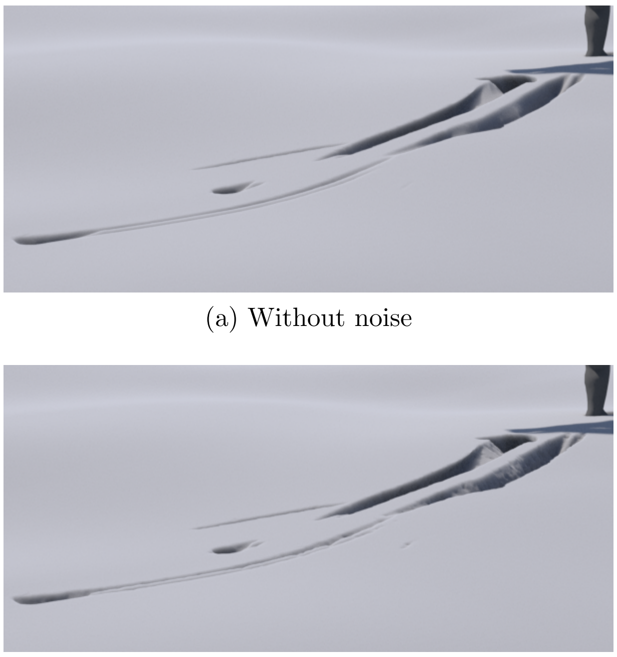
Performing multiple substeps can reduce this problem. However, a large number of substeps is required for animations with fast moving objects (see Fig. 3(b)), which greatly increases computational time. To overcome this limitation, the dynamic objects' geometry is sampled at times $t + n * \frac{\Delta t}{N}$, where $n = 0, 1, 2, ..., N$, and N is the number of samples. These samples are individually voxelized, and then combined into a single level set using a union operation (see Fig. 4(b)). In order to fill the gap between the samples, a smoothing operation is performed on the level set. Because this operation generally shrinks the level set, a dilation is performed before the smoothing operation to compensate for the volume loss. Besides, this dilation operation also helps to better connect the samples. The result of these operations is a smooth level set spanning the whole animation path (see Fig. 4(c)). This outlines an advantage of our approach compared to methods relying on height fields: both the base layer and the volume swept by the dynamic objects share a unified representation that makes it easy to compute operations such as CSG union and subtraction as well as smoothing and dilation. It should be noted that the number of samples is much lower than the number of simulation substeps needed to obtain similar results (see Fig. 3(b) and (c)), thus making this approach much faster.

4.1 Imprint Refinement

In reality, snow imprints rarely have the exact shape of their "source" object. Some snow is slightly pushed during the impact, and some is taken off as the object moves away. Moreover, snow does not react as a completely homogeneous material and it often contains small air pockets. Leaving a sharp and precise imprint does not result in a realistic imprint. To improve the realism of our approach, the dynamic objects' level set is updated with controllable irregularities before it is subtracted from the base layer (see Fig. 4(d)). To do so, we use a technique inspired by the level set surface modeling operators of Museth et al. [14]. These operators deform a level set by advecting it with a velocity field. This vector field is aligned with the gradient field of the level set, which forces the surface to move along its normal. In our approach, the length of the displacement is determined using a noise function:

$$\mathbf{v}(\mathbf{x}) = \frac{\nabla\phi(\mathbf{x})}{\|\nabla\phi(\mathbf{x})\|} \cdot \max(\text{noise}(\mathbf{x}), 0).$$

To make sure the base layer will not enter the object, negative displacements are prevented. The level set of the objects is advected with the velocity field, resulting in the final surface (Fig. 4(d)). Such noise has been added to all of the examples shown in this paper, and its parameters have been determined experimentally to result in a reasonable size, considering the expected size of the snow clumps. Fig. 5 shows the improved realism of the base layer with such noise.


(a) Without noise

(b) With noise

Fig. 5 Comparison of the base layer rendered without noise (a), and with noise (b).

5 Snow Particles

At this point, the base layer models the basic shape of the imprints in the snow. However, it provides a very limited interaction with the dynamics objects. To better mimic the behavior of snow moved by these interactions, snow particles are used. While these particles interact with the base layer and the dynamic objects, they also interact with each others. Thus, they can be pushed, piled up, and even projected in the air, allowing for a more complex behavior that complements the imprints from the base layer.

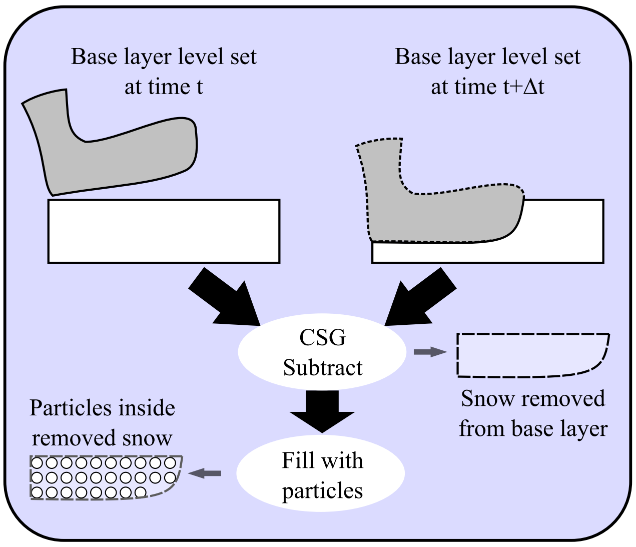

Base layer level set at time t
Base layer level set at time t+Δt
CSG Subtract
Snow removed from base layer
Particles inside removed snow
Fill with particles

Fig. 6 Snow particles sourcing. Snow compressed from the base layer is computed using a CSG subtract operation, and is then filled with snow particles.

5.1 Granular Simulation

We opted for a granular material simulation that relies on position based dynamics [10,13] as it can efficiently handle a lot of particles. Interactions between the snow particles and the base layer are handled by considering the base layer as a rigid body in the simulation. Furthermore, to the influence that gravity, collisions, attractions, and friction have on particles, we added the effect of air resistance:

$$v_i^* = \beta^{\Delta t} v_i,$$

where v_i is the particle's velocity, v_i^* is the modified velocity, and β is the air resistance coefficient in the range [0..1].

5.2 Particles Sourcing

As stated earlier, snow is not fully compressed when it collides with dynamic objects. This is considered in our approach by filling the volume of snow removed from the base layer with an amount of snow particles that is inversely proportional to the compression of the snow (see Fig. 6). This volume is computed by subtracting the base layer level set at time $t + \Delta t$ from that of time t using a CSG subtraction operation. The compression of the snow is controlled using a single parameter, the compression ratio, that varies between 0 (incompressible) and 1 (fully compressible). Given the number of particles that can fit inside the removed volume, this number is diminished as the compression ratio is increased. We consider that snow is initially at rest, therefore the particles are initialized with a null velocity. The whole particle sourcing operation is done only once per frame before handling the snow particles dynamics. Thus, snow particles are created at time t, and then from time t to $t + \Delta t$ the dynamic objects are animated while the particles are simulated. As the granular simulation is computed, the dynamic objects collide with the particles, letting the collision response determine the behavior and the velocity of these new particles.

5.3 Inactive Particles

In our approach, computation times and memory consumption are considerably lowered by using a signed distance field, i.e. the base layer, to depict snow that remained still. This effectively lowers the number of particles to process. Still, their number can greatly increase as more particles are added from the interaction between snow and dynamic objects. This can significantly affect the computation times. Because most particles quickly come to rest and stay still, a lot of computation time is wasted. To avoid such unnecessary calculations, only particles near dynamic objects or near moving particles are computed. To determine which particles should be simulated, a level set is built from the particles whose velocity is higher than a predefined velocity threshold and from the dynamic objects. Then, the distances of each particle to this level set is computed. If this distance is further than a target distance threshold, the particle is considered inactive. These inactive particles are not processed during the particle dynamics computations. It should be noted that they are still rendered, and that they are used when computing collision, attraction, and friction forces of active particles. They can also be reactivated later when dynamic objects or moving particles get closer. In our examples, the list of inactive particles is updated twice per frame to reduce the likelihood of fast moving dynamic objects leaving the simulated particles region. Another option would be to use a larger target distance threshold, and to update the list only once per frame. However this would include more particles that may not collide with the objects.

6 Snow Mist

Fig. 7 shows the three types of behavior modeled by our approach: the static base layer, moderate size snow clumps projected in the air (our snow particles), and the thin snow mist. Snow particles driven by the granular material simulation model pushed and projected

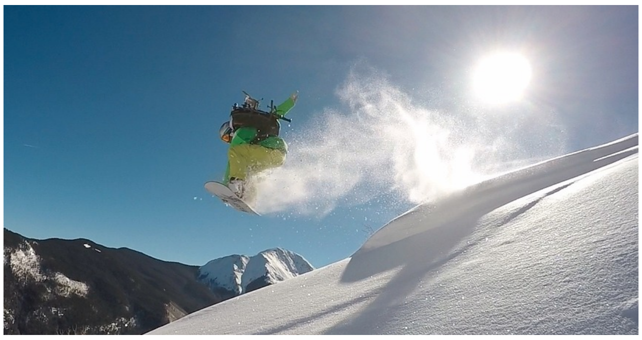


Fig. 7 In this photograph, snow mist is visible as tiny snow particles are projected in the air. This public domain image is provided by Adam Longnecker.

snow clumps quite well, but it would require a tremendous amount of tiny particles and long computation times to visually replicate the snow mist. Furthermore, snow mist requires a special treatment, as its behavior is highly affected by the surrounding air medium. The density distribution resulting from the turbulent motion typical of smoke simulations can be seen in the snow mist of Fig. 7. We therefore use another simulation paradigm to simulate snow mist in a more realistic way. Additionally, we have significant computation time gains by simulating this component separately from the others.

6.1 Snow Mist Simulation

We use an Eulerian incompressible fluid simulation for snow mist. Generally, in such simulation, gravity forces are not considered since the whole simulation domain is filled with gases of equivalent densities. However, since snow mist is much denser than the surrounding air, we apply gravity forces in the fluid simulation where the density is greater than 0. However, we use a lower value $g_{mist} = -2.0\ m/s^2$ in our examples to account for air friction.

6.2 Snow Mist Sourcing

Snow mist is sourced into the simulation from airborne snow particles, once per frame. To do so, potential snow particle candidates for sourcing are identified by testing their velocity against a threshold, to determine particles with high velocity. This criterion has been inferred from observations, and can be intuitively understood: higher velocity results in more energy and turbulence, which has a greater potential to break snow clumps into snow mist. In our examples the threshold value $v_{\text{mist}} = 2.0\ m/s$ is used. Additionally, falling snow particles, i.e. $\mathbf{g} \cdot \mathbf{v} > 0$, are ignored to prevent snow mist creation when particles fall due to gravity. Using the candidate particles, a seed density field is filled from the union of spheres at each particles' center (radius is twice that of the particles). To increase the realism at a marginal computation cost, we also rely on the addition of a limited amount of noise to this seed density field. Similarly, a seed vector field is filled from the average velocity of the neighbor particles, where seed density has been added. Finally, both the seed density field and the seed velocity field are added in the simulation.

Our observations showed that mass transfer between snow particles and the snow mist is limited. Furthermore, as the snow mist constantly dissipates in the mist simulation, the unaccounted mass gain is counteracted by this dissipation. Thus, mass transfer between the snow particles and the snow mist is not taken into account in our approach.

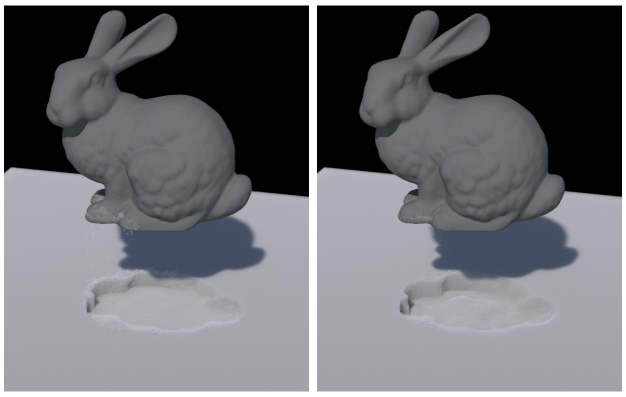

(a) $r_c = 0.0$ (b) $r_c = 0.75$

Fig. 8 Stanford bunny imprints. This scenario has been computed with no compression, i.e. with the compression ratio value $r_c = 0.0$ (a), and a high compression ratio $r_c = 0.75$ (b).

7 Rendering

Before being rendered, the base layer is tessellated, making it easy to use with any surface shader. However, as it is quite common to render snow as a density volume [19], the base layer is rendered as a volume with constant density. Similarly, particles are also rendered as a density volume. This volume is built using the union of spheres located at each particle's center, with each sphere having the same radius r as the particles. A smooth density falloff of length $r/2$ is added around each sphere to make snow look softer, as the density decreases at the snow boundaries. As for snow mist, its density field is rendered as it is.

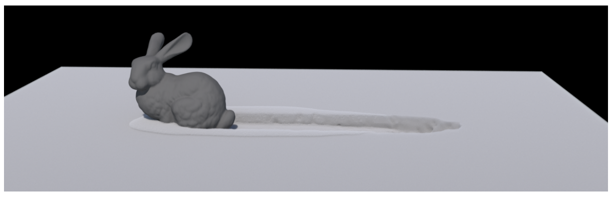
8 Implementation

Our approach has been integrated inside Houdini$^{\text{TM}}$. We used OpenVDB [15] to store level sets and apply operations on them, e.g. CSG, dilation, smoothing, and advection. The sparse representation used by this library helps to lower memory consumption and computation times considerably. The granular material simulator of Houdini$^{\text{TM}}$, which employs position based dynamics [10], was used to simulate snow particles. We extended their simulator to use our air friction model (see Sec. 5.1), instead of their force-based air friction. The snow mist solver is an adapted version of the smoke solver of Houdini$^{\text{TM}}$, to which our additional forces have been added. The solver uses a dynamic grid whose dimensions are adjusted based on the distribution of density in the scene. This greatly improved performances in our experiments. All our results were rendered using Houdini Mantra$^{\text{TM}}$. Their *uniform volume shader* was used for the base layer surface. Such a shader takes advantage of the assumption of a uniform density inside the mesh surface, which makes it faster than rendering an arbitrary density volume. Snow mist and snow particles density volumes are rendered using Houdini Mantra$^{\text{TM}}$ standard smoke shader *Billowy Smoke*. The density of the snow particles volume is scaled by a factor of 100 at render time to match the snow opacity from our observations.

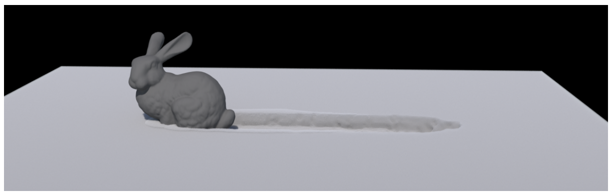
By relying on state of the art commercial software, our implementation demonstrates that the proposed approach is easy to integrate into current production pipeline tools. While we used Houdini$^{\text{TM}}$ in our implementation, the same base simulation paradigms are implemented in most popular commercial software. Therefore, our approach is not limited to a specific software vendor.

9 Results

We have tested our approach on a wide variety of scenarios. The animations for the examples shown in this paper are available in the accompanying video. In Fig. 8, the Stanford bunny is translated down and up in the snow creating an imprint. The contour of the bunny is well captured in the base layer level set. Furthermore, the noise applied on the base layer provides a more natural look at a negligible computation expense. This scenario has been computed with no compression in Fig. 8(a), and with a high level of compression in Fig. 8(b). Fewer particles are found in the imprints in the latter case, resulting in a deeper hole. In Fig. 9, the Stanford bunny pushes snow as it moves horizontally. Snow is transferred from the base layer to snow parti-



(a) $r_c = 0.0$



(b) $r_c = 0.75$

Fig. 9 The Stanford bunny slides to the left, pushing snow particles as it moves. This scenario was computed with no compression (a), and with a high level of compression (b).

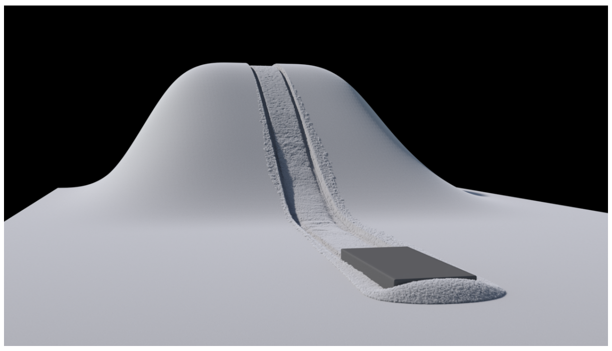


Fig. 10 A sleigh leaves tracks and pushes snow.

cles, and creates a pile of snow in front and beside of the bunny's path. These particles add the dynamics of snow not modeled by the base layer. As can be seen in Fig. 9(b), a smaller pile of snow is formed when using a higher compression ratio. Similarly, a sleigh leaves tracks in the snow in Fig. 10. Snow pushed by the sleigh shows a very high level of details. Our approach allows such level of details, even with a large simulation domain, by adding snow particles only where details are needed.

Our approach has also been validated with more complex animations, closer to what is found in an animation movie. In Fig. 11, a human character walks in the snow. As its feet move upward, snow particles are projected in the air, along with some snow mist. A gorilla walks, spins, and rolls in the snow in Fig. 12. All of its interactions with the snow are well captured by our approach. As snow particles are projected in the air, snow mist is automatically added to the fluid simulation, improving the realism, compared to methods relying only on height maps or granular material simulations.

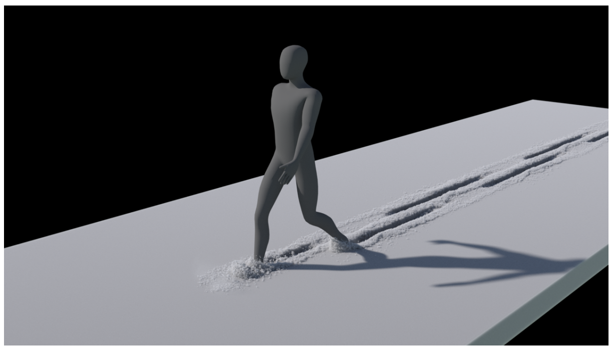


Fig. 11 A human character walking in the snow.

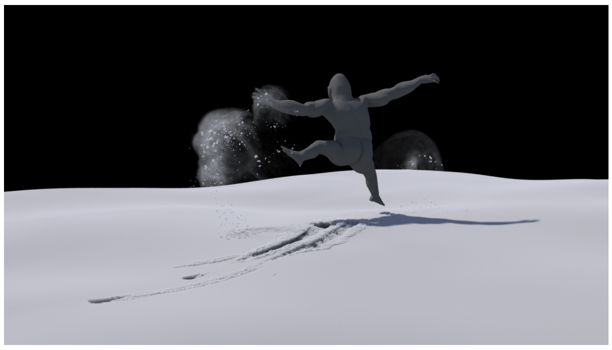


Fig. 12 A gorilla dances in the snow, leaving imprints as well as projecting snow and snow mist in the air.

The timings for all the examples shown in this paper are available in Table 1. Most of the time is spent in the snow particles simulation. The base layer is by far the fastest step in our approach. Thus, in production, it can be used to have an early feedback of the imprints in the snow, which can be used to adjust the animations quickly. We were able to lower considerably the computation time for the snow particles simulation by excluding inactive particles during computations (see Fig 13). Without such optimization, the dancing gorilla example required an average of $62.20s$ per frame to simulate snow particles. With the optimization, it only required an average of $39.25s$ per frame, a 1.58X speedup. All of our simulations were computed on a 6 CPUs Intel Core i7 with 56 GB of memory. The gorilla model was obtained from the TOSCA [3] project, and animated in Autodesk Maya® using a provided example animation setup.

Our approach considerably lowers memory consumption and computation times by using a level set, i.e. the base layer, to model snow that did not interact with the dynamic objects yet. We compared the memory consumption and timings of our approach with a particles-only simulation, where the whole snow volume was filled with particles, in the walking human

Example	Base layer cell size	Snow particle radius	# snow particles	Snow mist cell size	Average time per frame (s): Base Layer	Snow particles	Snow Mist	Total
Fig. 8(a)	0.005	0.00075	243,653	0.005	1.84	11.64	6.00	19.48
Fig. 8(b)	0.005	0.00075	60,759	0.005	1.84	4.36	5.09	11.29
Fig. 9(a)	0.005	0.00075	1,575,948	0.005	1.68	81.26	9.18	92.12
Fig. 9(b)	0.005	0.00075	396,272	0.005	1.73	15.90	6.18	23.81
Fig. 10	0.01	0.003	1,684,770	0.015	1.92	33.73	2.85	38.50
Fig. 11	0.0075	0.0025	879,062	0.01	3.15	11.30	6.65	21.10
Fig. 12	0.0075	0.0025	2,850,936	0.01	4.59	39.25	13.62	57.46

Table 1 Timings for all our examples.

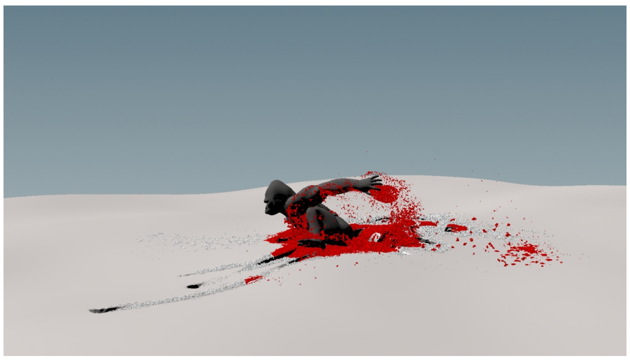


Fig. 13 Only active snow particles (in red), in the areas around dynamic objects and moving particles, are computed in our examples. This considerably lowers computation times.

character example. Because of memory limitations, a coarser resolution was used, setting the particles radius to $2r$. Even with the coarser resolution, the simulation required 6.9 GB of memory. Using a finer resolution, i.e. with the particles radius set to r, it would require approximately eight times more memory (55 GB). In contrast, the finer resolution simulation computed with our approach fits within 3.5 GB of memory for the base layer and the snow particles. As for the timings, the coarse particles-only simulation required on average $273.1s$ per frame. Using our inactive particles optimization, it could be lowered to $51.86s$ per frame. Still, even with the eight times finer resolution, our approach required on average only $21.1s$ per frame, even including the snow mist computation times.

Our approach has some limitations. The coupling between the snow and the dynamic object is unidirectional: the animated objects influence the snow, but the snow has no influence on the objects. Physically-based animation with locomotion controllers cannot be used with our method as it does not model the contact from the snow to the dynamic objects. Nevertheless, this is quite an advantage for typical visual effect animations, as it provides a perfect and predictable animation of the dynamic objects or characters. Even though we can capture a wide range of interactions, some cannot be captured. For example, as our approach assumes that snow remains static until it collides with an object, it is possible to design animations creating sharp snow cliffs and tunnels into which the surrounding base layer snow will never fall.

10 Conclusion

In this paper, we introduced an efficient workflow that handles the interaction of snow with dynamic objects. We decomposed the snow into three components that can be controlled individually. Firstly, the base layer can be controlled to influence the imprints' shape and details, at a low computational cost. Secondly, the snow particles control the complex dynamics of snow that gets pushed and thrown in the air. Finally, the snow mist provides control over the thinner airborne snow that behaves according to its surrounding air medium. Our approach lowers memory consumption and computation times by using the most expensive simulations only where needed. Additionally, the surface representation of the base layer allows the use of the same shader on the simulated snow as well as any background snow that would be left out of the simulation for efficiency reasons.

In the future, we would like to improve our model so that the base layer snow is transformed to snow particles in other situations such as along sharp slopes or when a dynamic object moves below the base layer surface. Another avenue for future work would be to improve the way compression is handled using a physically-based model that could consider properties such as air temperature, snow temperature, and snow wetness. We would also like to create a model that scales the drag forces according to the density of snow particles to better mimic the friction of snow with the air. Also, our approach could be modified to convert back static snow particles to the base layer. We believe such improvement could lower the memory consumption. However, care must be taken to avoid sudden jumps or flickering of the surface. A possible solution would be to convert only hidden layers of particles, i.e. particles that are fully covered, similarly to Zhu and Yang [25] and Wong et al. [23]. Nevertheless, it should be noted that such modification would prevent the user from running the entire base layer simulation before simulating snow particles.

Acknowledgements

This work was funded by Mokko Studio, Mitacs, Prompt, and École de technologie supérieure. We would like to thank employees of Mokko Studio for their feedback throughout the project. We would also like to thank SideFX^{TM} for providing Houdini^{TM} licenses for research. We would like to thank Bruno Roy as well for the walk animation. Lastly, we thank the anonymous reviewers for their thoughtful comments and suggestions.

References

1. Alduán, I., Otaduy, M.A.: SPH granular flow with friction and cohesion. In: Proc. of ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 25–32 (2011)
2. Bell, N., Yu, Y., Mucha, P.J.: Particle-based simulation of granular materials. In: Proc. of ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 77–86 (2005)
3. Bronstein, A., Bronstein, M., Kimmel, R.: Numerical Geometry of Non-Rigid Shapes. Springer (2008)
4. Fearing, P.: Computer modelling of fallen snow. In: Proc. of SIGGRAPH 2000, Annual Conference Series, pp. 37–46 (2000)
5. Feldman, B.E., O'Brien, J.F.: Modeling the accumulation of wind-driven snow. In: SIGGRAPH 2002 Conference Abstracts and Applications, pp. 218–218. ACM (2002)
6. Festenberg, N.v., Gumhold, S.: A geometric algorithm for snow distribution in virtual scenes. In: Eurographics Workshop on Natural Phenomena, pp. 15–25 (2009)
7. Festenberg, N.v., Gumhold, S.: Diffusion-based snow cover generation. Computer Graphics Forum **30**(6), 1837–1849 (2011)
8. Ihmsen, M., Wahl, A., Teschner, M.: A lagrangian framework for simulating granular material with high detail. Computers & Graphics **37**(7), 800–808 (2013)
9. Lenaerts, T., Dutré, P.: Mixing fluids and granular materials. Computer Graphics Forum **28**(2), 213–218 (2009)
10. Macklin, M., Müller, M., Chentanez, N., Kim, T.Y.: Unified particle physics for real-time applications. ACM Trans. Graph. **33**(4), 153:1–153:12 (2014)
11. Maréchal, N., Guérin, E., Galin, E., Mérillou, S., Mérillou, N.: Heat transfer simulation for modeling realistic winter sceneries. Computer Graphics Forum **29**(2), 449–458 (2010)
12. Moeslund, T.B., Madsen, C.B., Aagaard, M., Lerche, D.: Modeling falling and accumulating snow. In: Vision, Video and Graphics, vol. 2 (2005)
13. Müller, M., Heidelberger, B., Hennix, M., Ratcliff, J.: Position based dynamics. Journal of Visual Communication and Image Representation **18**(2), 109–118 (2007)
14. Museth, K., Breen, D.E., Whitaker, R.T., Barr, A.H.: Level set surface editing operators. ACM Trans. Graph. **21**(3), 330–338 (2002)
15. Museth, K., Lait, J., Johanson, J., Budsberg, J., Henderson, R., Alden, M., Cucka, P., Hill, D., Pearce, A.: OpenVDB: an open-source data structure and toolkit for high-resolution volumes. In: SIGGRAPH 2013 Courses, p. 19. ACM (2013)
16. Narain, R., Golas, A., Lin, M.C.: Free-flowing granular materials with two-way solid coupling. ACM Trans. Graph. **29**(6), 173:1–173:10 (2010)
17. Onoue, K., Nishita, T.: An interactive deformation system for granular material. Computer Graphics Forum **24**(1), 51–60 (2005)
18. Pla-Castells, M., García-Fernández, I., Martinez-Dura, R., et al.: Physically-based interactive sand simulation. In: Eurographics 2008, Short Papers, pp. 21–24 (2008)
19. Stomakhin, A., Schroeder, C., Chai, L., Teran, J., Selle, A.: A material point method for snow simulation. ACM Trans. Graph. **32**(4), 102 (2013)
20. Sumner, R.W., O'Brien, J.F., Hodgins, J.K.: Animating sand, mud, and snow. Computer Graphics Forum **18**(1), 17–26 (1999)
21. Takahashi, T., Fujishiro, I.: Particle-based simulation of snow trampling taking sintering effect into account. In: SIGGRAPH 2012 Posters, p. 7. ACM (2012)
22. Wang, C., Wang, Z., Xia, T., Peng, Q.: Real-time snowing simulation. The Visual Computer **22**(5), 315–323 (2006)
23. Wong, S.K., Fu, I., et al.: Hybrid-based snow simulation and snow rendering with shell textures. Computer Animation and Virtual Worlds **26**(3-4), 413–421 (2015)
24. Zeng, Y.L., Tan, C.I., Tai, W.K., Yang, M.T., Chiang, C.C., Chang, C.C.: A momentum-based deformation system for granular material. Computer Animation and Virtual Worlds **18**(4-5), 289–300 (2007)
25. Zhu, B., Yang, X.: Animating sand as a surface flow. Eurographics 2010, Short Papers (2010)
26. Zhu, Y., Bridson, R.: Animating sand as a fluid. ACM Trans. Graph. **24**(3), 965–972 (2005)