Eurographics Symposium on Parallel Graphics and Visualization (2022)
R.. Bujack, J. Tierny, F. Sadlo (Editors)

Iterative discrete element solver for efficient snow simulation

Prashant Goswami[†][*] Adrian Nordin[*] Simon Nylén[*]

Blekinge Institute of Technology, Sweden

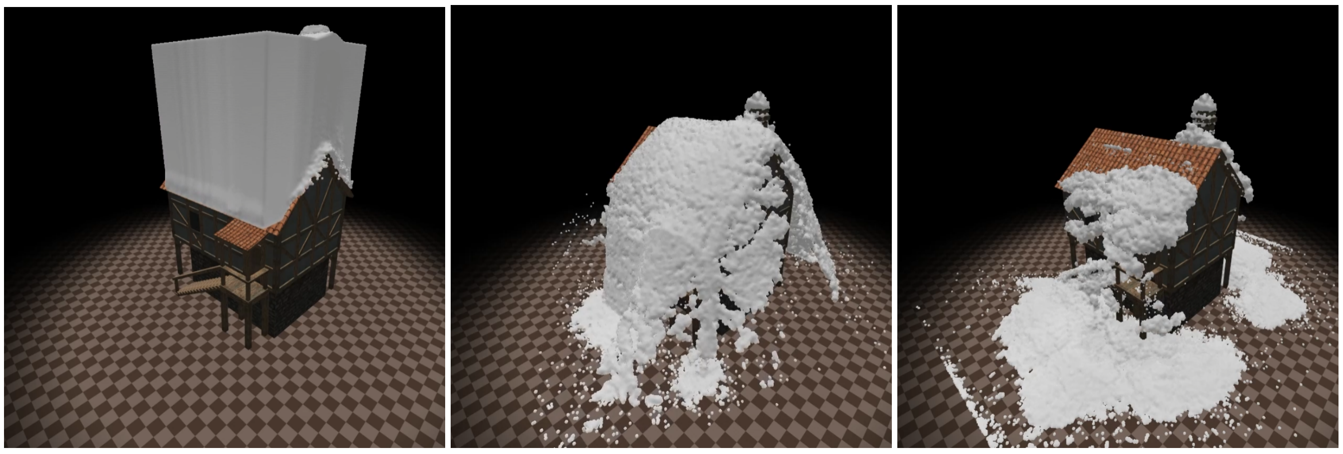


Figure 1: *1M wet snow particles simulated and interacting with a complex geometry consisting of 323K static boundary particles at a time step of 1 ms using our GPU-based method, supporting both simulation and visualization at 20 fps.*

Abstract

This paper presents a novel Discrete Element Method (DEM) on the GPU for efficient snow simulation. To this end, our approach employs an iterative scheme on particles that easily allows the snow density to vary vastly for simulation while still maintaining a relatively large time step. We provide computationally inexpensive ways to capture cohesion and compression in the snow that enables us to generalize the behavior of various kinds of snow (like dry, wet, etc.) by varying physical parameters within the same simulator. We achieve a speed-up of nearly eight times with one million snow particles over the existing real-time method, even while dealing with scenes containing complex object boundaries. Furthermore, our simulator not only retains stability at these large time steps but also improves upon the physical behavior of the existing method. We have also conducted a user evaluation of our approach, where a majority of the participants voted in favor of its realism value for computer games.

CCS Concepts
• ***Computing methodologies*** $\rightarrow$ ***Physical simulation;***

1. Introduction

Physics-based snow simulation is a challenging task. This is because snow is a complex material involving a multitude of factors that affect its behavior, such as cohesion, compression, and thermodynamics. At the same time, snow forms an important component in many video games (Red Dead Redemption 2, Snow Runner etc.) and other such applications. Recently offline methods have been developed in computer graphics (CG) that can achieve impressive snow simulation. This includes both Eulerian [SSC*13] and Lagrangian [GHB*20] approaches. However, most of the existing implementations come with the constraint of a high computational burden, taking anywhere from a few seconds to even a few minutes to finish a single frame. This renders them unsuitable for use in any real-time applications. Therefore, it is common that the behavior of snow in such applications is usually captured with the help of mesh deformation or other procedural methods instead.

Efficient simulation methods are capable of capturing vari-

[†]prashant.goswami@bth.se
[*]all authors have an equivalent contribution

DOI: 10.2312/pgv.20221062

ous physical phenomena in CG. This includes fluids, gases, deformable and rigid bodies (including sand) among other materials [MMCK14, MMC*20]. Large time steps for divergence-free fluid simulation are shown to accelerate computation in [BK17]. GPU porting has demonstrated the positive aspect of speeding up simulations that are inherently parallel in their nature [GEF15,GSSP10]. However, with the exception of Goswami et al. [GMH19], no other method has demonstrated the dynamic behavior of snow for CG purposes in real-time computation. However, a key drawback of their method is the low time step it is constrained to take. This necessitates running several folds more of physics iterations, thereby affecting the overall efficiency adversely.

In this paper, we present a novel particle-based, efficient snow DEM simulation method on the GPU. To this end, the proposed method employs an iterative solver, which is inspired by the predictive-corrective scheme in [SP09] in principle, albeit with significant differences. The main contributions of our method are:

- To present an efficient DEM iterative snow simulator on the GPU that can handle reasonably large time steps while maintaining stability, and real-time to interactive frame rates.
- To efficiently capture the snow compression and bonding behavior, thereby allowing behavior ranging from the soft snow to that of the hard snow (dry, wet) and tending to the ice.
- To efficiently incorporate boundary handling with solid objects, thereby allowing complex snow-object interactions through two-way coupling.
- To validate the visual realism value of our approach with the help of a user study.

We achieve a significant speed-up over the existing real-time method. For one million snow particles together with over a quarter of a million of boundary particles, our iterative framework is nearly 8 times faster than the base simulator [GMH19] and the simulation is still interactive. Furthermore, our simulation improves upon the behavior of this base simulator. With the help of static boundary particles, our solver can easily handle complex snow-object interactions and two-way coupling, which is a crucial requirement in most games and other similar applications.

The presented technique could be suitable for games and other virtual applications, where a certain amount of accuracy can be traded-off for the sake of efficiency. Furthermore, its efficiency and simplicity make integration easier with the other dynamic components in the game world. In order to sustain a high frame rate, we have introduced certain approximations in our solver. To this end, we validated the proposed technique through a perceptual evaluation consisting of 31 participants who found the visual value of our approach a good fit for the computer games. Furthermore, we provide detailed exposition and analysis of our results for the proposed method.

The remainder of this paper is organized as follows. Sec. 2 discusses the existing related work in the field. In Sec. 3, we first briefly introduce the foundations of the base method, leading to a detailed exposure of the proposed approach in Sec. 4. Sec. 5 covers the remaining implementation details together with a thorough exposition of the visual results, comparison with the base method, and other related analyses, including the user evaluation.

2. Related Work

Snow is frequently simplified in real-time applications, and often captured using procedural methods like heightmaps [DGP16,TF12, CEG*18]. Heightmaps or similar techniques can be also be used to simulate different stages of snow accumulation [HM09, FG11, RLD15, JP20]. Particle-based methods for snow animation have also been developed in CG. Research on real-time particle-based methods [FZ12, TZWZ09, YLJ11] explore enhancements of realistic virtual 3D scenes by simulating snowfall. These simulations explore snow at a particle-level compared to heightmaps, which only simulate the surface of the snow. However, only the external forces such as gravity and wind are accounted for, not the critical internal factors such as cohesion between the particles or the compressibility of the snow particles. Additional parameters need to be considered to further increase the immersion of games and other real-time applications incorporating snow. These simulations are computationally expensive and therefore, often cannot execute in real-time.

Gissler et al. [GHB*20] present a feature-rich snow simulation approach based on an implicit compressible SPH solver to simulate parameters such as deformation, breaking, compression, and phase transition. The simulator can handle large time steps and a more extensive set of scenarios than the hybrid Eulerian/Lagrangian methods, specifically smaller volumes of snow. These smaller volumes allow the simulator to capture both snowfall and accumulation in combination with the scenarios that the hybrid Eulerian/Lagrangian methods allow. Hybrid structure in [WF15] employs a combination of particles and springs to model and simulate brittle snow. Stomakhin et al. [SSC*13] propose a hybrid Eulerian/Lagrangian Material Point Method (MPM) which is used to simulate snow. This simulation captures many snow parameters that allow for a large set of different snow behaviors, ranging from dry and powdery to wet and compressed snow. MPM is temporally adapted using regional time stepping in [FHHJ18] and ported on GPU in [WQS*20]. These papers further establish that impressive snow interactions can be simulated using particles, albeit not in real-time.

DEM has also been used to capture physically accurate behaviors of granular materials. Mao et al. [MWXC04] utilize DEM with particles represented as spheres to simulate particle damping using efficient collision detection. Their simulation results are validated by physical experiments, which are used for comparison. Granular materials like sand have also been simulated [AO11, IWT12] with the help of predictive-corrective incompressible SPH (PCISPH) method [SP09].

Macklin et al. [MMCK14] have proposed a unified real-time simulator (*FleX*) with capabilities of simulating materials such as gases, liquids, deformable solids, rigid bodies, and cloth. Their framework implements several materials in real-time that were previously implemented in offline solutions because of their complexity; however, snow is not one of them. This framework is a part of Nvidia's physics engine *PhysX*, which has been incorporated into multiple games [NVI14]. Machine learning (ML) has recently been used to accelerate the physical simulations [SGGP*20, TKC21] in CG. However, most of these ML methods require a base simulator whose behavior is learnt and approximated by the neural network. Therefore, these ML methods can benefit from a more efficient or evolved physics-based solver.

Hagenmüller et al. [HCN15] model hardened snow under deformations with a granular description using DEM. The research models uniaxial confined compression while using multiple deformation phases to determine the elastic or plastic behavior of the snow. This approach was successfully adopted in [GMH19], wherein snow can be simulated in real-time for low to medium particle counts with some physical approximations. However, a major limitation of their technique was the constraint of using a small time step to maintain stability. Our work aims to provide a simulator incorporating complex snow behavior, which could be useful in games and other real-time applications.

3. Base method

In this section, we briefly recap the fundamentals of the base method in [GMH19]. This base method employs a simple DEM particle-based approach as against SPH kernel interpolation. Each particle has a mass m and radius r, that lies in the range of $r_i \le r \le r_s$. A particle with radius r_s is considered to be composed of pure snow ($\rho_s = 100kg/m^3$), whereas with r_i as pure ice ($\rho_i = 900kg/m^3$). Starting with the snow state as the particle loses its entrapped air, its state is a mix containing η proportion of snow and (1-η) proportion of ice, while still retaining the initial mass. η itself is determined with the help of the current particle radius as $\eta = \frac{r-r_i}{r_s-r_i}$. All the physical properties of a particle (Young's modulus Y, normal cohesion σ) are computed by linearly interpolating them on the properties of soft snow and ice, using determined η.

The basic steps involved in the base method are laid out in Alg. 1. The neighborhood set is determined similar to [Gre10] using particle index hashing on a virtual grid. Only particles touching the particle in question constitute its neighborhood set. This reduces the number of neighbors for each particle drastically when compared to SPH. The base method extends the cohesion model provided by Hagenmüller [HCN15], which focuses only on the hardened snow. This extension allows the particles to range between soft snow and ice by interpolating properties based on the densities of particles, as explained above.

Air drag on each particle is computed using the standard air drag force equation [WR27] (Eq. 1), where $C_d = 0.05$ is the drag coefficient, $\rho = 1.4$ is the density of air, $\vec{\boldsymbol{v}}$ is the particle's velocity and A its surface area.

$$\vec{f_{air}} = -\frac{1}{2} C_d \rho (\vec{\boldsymbol{v}} \cdot \vec{\boldsymbol{v}}) A \frac{\vec{\boldsymbol{v}}}{|\vec{\boldsymbol{v}}|} \tag{1}$$

The cohesive normal force, based on a linear spring model following Hooke's law, is applied along the contact normal between interacting snow particles p and q, which keeps the snow particles together and can be seen in Eq. 2.

$$\vec{\boldsymbol{f}}_n = \begin{cases} -\frac{Y_p r_p + Y_q r_q}{2} \delta \vec{\boldsymbol{n}} & -\frac{Y_p r_p + Y_q r_q}{2} \delta < 4 \frac{\sigma_{np} r_p^2 + \sigma_{nq} r_q^2}{2} \\ 0 \text{ (Cohesion Broken)} & otherwise \end{cases} \tag{2}$$

The formula is dependent on the particle overlap δ, radius r, Young's modulus Y, contact normal $\vec{\boldsymbol{n}}$ and the normal cohesion σ_n. The cohesive force can be either attractive or repulsive depending on the overlap (δ) between particles being negative (touching or in vicinity) or positive (penetrating), respectively. When the force exceeds the maximum force threshold, cohesion is broken, and no force is applied. In addition, the particles are influenced by a frictional shear, which gives cause for a tangential force. The tangential force is formulated in Eq. 3.

$$\vec{\boldsymbol{f}}_{\boldsymbol{t}} = (\vec{\boldsymbol{u}}/|\vec{\boldsymbol{u}}|)|\vec{\boldsymbol{f}}_{\boldsymbol{n}}| tan(\varphi) \tag{3}$$

The angle of repose φ is the steepest angle a pile can form for granular material and $\vec{\boldsymbol{u}}$ is the shear displacement. For the sake of simplifying computation, cohesive bonds are created based only on the normal force. The bonds between particles determine the application of cohesion; if a particle has bonds, it attracts other particles. Otherwise, the particle is not affected by any attraction to the other snow particles.

Algorithm 1 Base Snow Simulation Method

```
 1: while animating do
 2:     for all snow particle p do
 3:         Find Neighbourhoods $N_p(t)$
 4:     for all snow particle p do
 5:         Apply GravitationalForce(t)
 6:         Apply AirDrag(t)                        ▷ Eq. 1
 7:         Compute CohesionForces(t)               ▷ Eq. 2
 8:         Compute TangentialForces(t)             ▷ Eq. 3
 9:         Compute Compression(t)                  ▷ Eq. 4
10:         ComputeThermodynamics(t)
11:     for all snow particle p do
12:         Update velocity $\vec{v}(t+\Delta t)$
13:         Update position $\vec{x}(t+\Delta t)$
```

Snow volumes contain air pockets, and these pockets allow snow to be compressed. The summation of every uniaxial opposite force acting on the particle is used to compute compressive force (Eq. 4). Since snow is an elastoplastic material, the shape will return to its original form unless a force threshold $F_{threshold}$ is reached at which the particles experience plastic deformation. To reflect this, a durability coefficient d (see Eq. 5) is implemented similar to Takahasi et al. [TFN14] which ranges between 0 and 1.

$$f_c = \sqrt{min(\vec{\boldsymbol{f}}_{+x}, \vec{\boldsymbol{f}}_{-x})^2 + min(\vec{\boldsymbol{f}}_{+y}, \vec{\boldsymbol{f}}_{-y})^2 + min(\vec{\boldsymbol{f}}_{+z}, \vec{\boldsymbol{f}}_{-z})^2} \tag{4}$$

f_c is the compression force, $\vec{\boldsymbol{f}}_+$ and $\vec{\boldsymbol{f}}_-$ is the compressive forces acting in the positive and negative directions of the base axes, respectively. The particle durability coefficient, d, is interpolated between snow and ice density based on the pressure p_c and durability change coefficient k_q (Eq. 5).

$$d = \begin{cases} (\rho - \rho_{ice})/(\rho_{snow} - \rho_{ice}) - k_q p_c & f_c > f_{threshold} \\ \text{d (Unchanged)} & otherwise \end{cases} \tag{5}$$

To determine the magnitude of the threshold at which snow particles start to experience the plastic deformation, Eq. 6 is used.

$$f_{threshold} = f_{minW} + \frac{e^{\frac{\rho}{100}-1} - c_1}{c_2} f_{maxW} \tag{6}$$

where f_{minW} is the minimum magnitude of force a particle can withstand prior to initial compression, $c_1 = 0.00035$ and $c_2 = 2980.96$. The term d is linearly transformed to reduce the particle radius with compression.

An approximate bonding behavior between snow particles is captured by introducing the quantity bond threshold $\phi = \frac{n_{curr}}{n_{max}}$, where n_{curr} is the current neighbor count of a particle and n_{max} is the maximum count it had so far. The thermodynamics framework from Iwasaki et al. [IUDN10] is also incorporated in the base method. Our work does not modify the thermodynamics applied; hence in the remainder of this paper, we focus on the physical components that we have extended. We refer the reader to [GMH19] for a more detailed insight into the base method.

4. Our method

Our method proposes a DEM-based technique that applies an iterative scheme to correct particle positions in the snow physics solver. However, our technique varies with respect to the existing efficient physics-based solvers in CG with respect to a few essential aspects:

- Firstly, almost all solvers recently introduced in CG (divergence-free SPH [BK17], implicit incompressible SPH [ICS*14], position-based fluids [MM13], predictive-corrective incompressible SPH [SP09]), fix the SPH density of particles iteratively in each loop, where the density is computed in the standard SPH manner by taking a smoothed value over its neighbors within the support radius. The same is also true for other similar solvers, for example, involving snow [GHB*20] and sand [AO11]. In contrast to these methods, our DEM solver accounts for the density of each particle individually based on its size (radius) or compression level (as explained in Sec. 3) and iterates to minimize the overlap between each particle and its neighbors.
- Secondly, we, therefore, do not need to employ interpolation kernels to compute interaction forces between snow particles, which is also consistent with the original DEM approach of Hagenmüller et al. [HCN15].
- Thirdly, we incorporate factors such as bonding between snow particles and compression as a part of our solver. Due to compression, our simulation consists of particles that can vary in their radii by a factor of 2. Furthermore, we have designed an alternative error metric that considers the compressibility of a particle to compute its overlap error.

We sample the interacting boundaries with virtual static ice particles for the sake of the snow-boundary interaction. This helps us to reproduce the behavior of the solid boundaries effectively without introducing a different material or force. However, in order to exclude the boundary particles from any temperature-based changes such as melting, they are excluded from the thermodynamics.

Alg. 2 lists all the steps involved in our solver. The solver essentially minimizes the overlap between snow particles as they are advanced in the simulation when subjected to various forces. The proposed method begins with the neighborhood computation for the snow particles (line 3), wherein both the neighboring snow and the boundary particles are determined. The external forces ($\vec{f}_p^{ext}$) for particle p, comprising gravity and air drag, are determined once per frame for each particle outside the iterative loop (line 5). In addition to this, the compression force ($\vec{f}^{+,-}$) and the total accumulated force is also initialized per particle ($\vec{f}_p^{acc}$) (line 6-7).

For each animating frame, we warm start our solver by computing the adhesion, normal, tangential, and plastic forces (coefficient of restitution) for each snow particle with its neighboring boundary particles, if present (line 9). Similarly, the normal and the tangential forces are gathered from all the snow neighbors (line 10), followed by computation of the corresponding compression force (line 11). It is worthwhile noting that in PCISPH, this warm start is not done, and all values are initialized to 0.

We then step into the iterative-relaxation loop (line 12), wherein an estimate of velocity and position (lines 14-15) is obtained with the help of accumulated and external forces. This, in turn, gives us the current estimate of the snow particle positions, and their mutual overlaps thereof. The next step in iteration entails gathering neighboring boundary forces (line 17), and the normal forces with all touching snow neighbors for each snow particle (line 18), barring the tangential forces. The compression forces corresponding to each snow particle are updated after updating $\vec{f}^{acc}$ (line 19). This is followed by a global computation of the current maximum overlap between any two snow particles in the simulation (line 21) using a parallel max reduction operation.

The global iterations are continued till the overlap error drops below a certain threshold (5% in our case), given the constraints of a minimum (1) and a maximum number of iterations (3). The final velocity is obtained by adding together the computed external and accumulated forces ($\vec{f}_p^{ext,acc}$) for each particle, which is then used to update the particle position. The external objects interacting with the snow particles are moved at the end of each frame.

In the following, we explain all the relevant forces and factors involved in our solver in more detail.

4.1. Forces

Tangential force: The tangential force resulting from the snow-snow or snow-boundary interaction is a crucial factor for the simulation stability. However, it is not computed in the iterative loop and hence not gathered in $\vec{f}^{acc}$ (line 17-18), unlike the normal force. The reason for this separation is that the tangential force, as derived by Hagenmüller [HCN15], is not stable at large time-steps. An additional reason is that when the prediction is performed, the particles collide with a much larger set of particles than their initial or final neighboring set. If the tangent force is accumulated like the other forces, the particle behaves unexpectedly and potentially reacts to particles that it never actually collided with. Our experiments suggest that the tangential force computed in the initialization phase (lines 9-10) provides us with the best stability. Please note that the initialized tangential force plays a vital role in stabilizing the particles during the accumulation of the normal force (lines 17-18).

Normal force: As observed before, $\vec{f}_n$ can either be repulsive or

Algorithm 2 Iterative Snow Solver

```
1: while simulating do
2:     for all snow particle $p$ do
3:         find $N_p(t)$
4:     for all snow particle $p$ do
5:         calculate external forces $\vec{f}_p^{ext} = \vec{f}_p^{g,air}(t)$
6:         initialise compression forces $\vec{f}_p^{+,-} = \vec{0}$
7:         initialise accumulated force $\vec{f}_p^{acc} = \vec{0}$
8:     for all snow particle $p$ do
9:         compute boundary forces $\vec{f}_p^{acc} += \vec{f}_p^{ad,n,t,cor}(t)$
10:        compute snow forces $\vec{f}_p^{acc} += \vec{f}_p^{n,t}(t)$
11:        update compression forces $\vec{f}_p^{+,-}(t+1)$
12:    while $(overlapError > 5\%$ and $iter \leq minIter)$ do
13:        for all snow particle $p$ do
14:            estimate velocity $\vec{v}_p^*(t+1) = \vec{v}_p + (\vec{f}_p^{ext,acc}/m_p) \cdot \Delta t$
15:            estimate position $\vec{x}_p^*(t+1) = \vec{v}_p^*(t+1) \cdot \Delta t$
16:        for all snow particle $p$ do
17:            compute boundary forces $\vec{f}_p^{acc} += \vec{f}_p^{ad,n,cor}(t+1)$
18:            compute snow forces $\vec{f}_p^{acc} += \vec{f}^n(t+1)$
19:            update compression forces $\vec{f}_p^{+,-}(t+1)$
20:        for all snow particle $p$ do
21:            $overlapError = \mathtt{max}(overlapError, overlap_p)$
22:        $iter = iter + 1$
23:    for all snow particle $p$ do
24:        update bonds $B_p$
25:        compute compression $\vec{f}_p^{+,-}$
26:    for all snow particle $p$ do
27:        integrate velocity $\vec{v}_p(t+1) = \vec{v}_p(t) + (\vec{f}_p^{ext,acc}/m_p) \cdot \Delta t$
28:        integrate position $\vec{x}_p(t+1) = \vec{v}_p(t+1) \cdot \Delta t$
29:        $\vec{x}_p(t+1) = \mathtt{OscillationControl}\ (\vec{x}_p(t+1), \vec{x}_p(t))$
```

attractive depending on if the particles are mutually overlapping or are in close proximity to each other, respectively (see also Eq. 2). The normal force $\vec{f}_n$ is accumulated in $\vec{f}_{acc}$ (lines 9-10, 17-18) till the overlap error between any two snow particles in the simulation falls below a certain threshold (line 12). This accumulation occurs both when the nature of force is attractive or repulsive. However, $\vec{f}_n$ computed in Eq. 2 is multiplied with a normal force factor to stabilize the system since we are using larger time-steps than the base method. This factor was introduced to provide the additional force required to lower the overlaps and enable a faster convergence during the iterations. To this end, an empirical value of 3 was found to give a satisfactory behavior for this factor.

4.2. States of snow

Unlike the base method, our solver can achieve varied behavior of different forms of snow. Stomakhlin et al. [SSC*13] generate this effect with the help of several parameters that are directly involved in their equations. In our case, this is made possible by defining three important parameters involved in the iterative algorithm, corresponding to various kinds of snow (dry snow, wet snow, and ice, respectively):

- Young's modulus $\rightarrow Y_{ds}$, Y_{ws}, Y_i
- Normal cohesion factor $\rightarrow \sigma_{ds}$, σ_{ws}, σ_i
- Bond threshold $\rightarrow \phi_{ds}$, ϕ_{ws}, ϕ_i

Given a particular state of snow (dry or wet), a particle in Eq. 2 interpolates between corresponding parameters of snow (Y_{ds}, σ_{ds} or Y_{ws}, σ_{ws}) and those of ice. As stated before, this interpolation is performed based on the proportion of snow (η) and the proportion of ice (1-η) in a particle, based on its radius. The normal cohesion factor applies in conjunction with Young's modulus to determine the elastic properties of the particles. Different values for these parameters enabled the proposed method to alter the behavior of the snow depending on the snow type.

Bonds: In order to reproduce a more realistic behavior, the strength of bonding between the snow particles in different states is pivotal to capture. In addition to Y and σ, another essential parameter to capture the effect of wet and dry snow is the bond threshold ϕ. This parameter is crucial since the ease with which bonds between snow particles break has an important effect on reproducing the visual behavior of different types of snow. Ideally, each particle should track its neighbor history to bookkeep the existing and broken bonds. This process is very demanding both computationally and memory-wise. We instead employ the approximation devised in the base method, wherein a sudden loss of neighbors is attributed to breaking of bonds and is tracked with the help of $\phi = \frac{n_{curr}}{n_{max}}$. However, we determine different bond threshold values corresponding to different states of snow (ϕ_{ds}, ϕ_{ws}, ϕ_i). In Alg. 2, the broken bonds are updated using this ϕ after the iteration process is completed (line 24).

Compression: The effect of compression on snow is visible during self-compression and interaction with other objects. In order to compute the compression of each particle, positive and negative components of the accumulated force corresponding to each axis need to be maintained separately (Eq. 4). These components are plugged in Eq. 4 to calculate the compressive force, and then compared to $f_{threshold}$ (Eq. 6) to determine the change in durability d. Our experiments suggest that in our iterative approach, the effect of compression is best captured once the final forces on all particles are determined and not iteratively (line 25). It is for this reason that the particles are compressed after stepping out of the iteration process. It is important to note that the physical properties of particles (Y, σ, ϕ) change with compression, which in turn changes the visual appearance of the snow undergoing simulation.

Error metric: The overlap computation between any two snow particles p and q is modified based on the softness of snow particle in our solver and calculated using Eq. 7.

$$overlap = (r_p + r_q - ||\vec{x}_p - \vec{x}_q||)\Upsilon \tag{7}$$

Here the radius of each particle is r_p and r_q respectively, and $||\vec{x}_p - \vec{x}_q||$ is the distance between the center of particles. Since it is possible for the snow particles to be compressed in contrast to the ice particles that have been compressed to their limit, an adjustment factor $\Upsilon = 0.1 - 0.9\eta$ has been introduced. This factor ensures that a fully compressed ice particle (with radius r_i) represents the complete overlap, while a pure snow particle with (radius r_s) only accounts for 10% of its actual overlap. It also linearly interpolates any intermediate compression state based on these two extremes. This prevents the iterative loop from triggering because of the snow particles that are likely to be further compressed.

4.3. Boundary forces

In our implementation, the boundary force obtained from static boundary particles plays a vital role in maintaining simulation stability at relatively large time steps.

Adhesion: In the proposed method, with the addition of boundary objects, an adhesion force was introduced, acting in the normal contact direction. Different from cohesion, adhesion acts between particles of dissimilar materials. By having an adhesive force on the boundary objects, the snow particles can stick to surfaces. Similar to Gissler et al. [GHB*20], we use the adhesion model proposed by Akinci et al. [AAT13] which formulate the adhesive equation as follows in Equation 8.

$$\vec{f}^{\,adhesion}_{p\leftarrow b} = -\beta m_p \Psi_{b_b} A(|\vec{x}_p - \vec{x}_b|)\frac{\vec{x}_p - \vec{x}_b}{|\vec{x}_p - \vec{x}_b|} \tag{8}$$

Here subscript b symbolises the boundary particles and p the snow particle, $\vec{x}$ is the centre position of particles, β stands for the adhesion coefficient, m is the mass, Ψ_b is the volume of the boundary particle, and A is the modified spline function in Eq. 9, where $h = 2r_s$.

$$A(r) = \frac{0.07}{h^{3.25}} \begin{cases} \sqrt[4]{-\frac{4r^2}{h} + 6r - 2h} & 2r > h \wedge r \leq r \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

Friction: In addition to the adhesion force, snow particles are also influenced by a tangential force when colliding with the boundary particles. The frictional shear force in Equation 3 is adapted for the snow-boundary interactions by replacing the tangent of the repose angle $tan(\varphi)$ with a coefficient of friction μ. The frictional shear force is only applied if there is an overlap. It is important to note that while the adhesion force is normal in its nature, the frictional force is tangential.

Restitution: The introduction of static boundary particles necessitates an additional force to prevent the snow particles from ricocheting off the floor. This instability occurrs due to the same normal force computation for snow-boundary interactions as the snow-snow interactions, giving the collisions a spring-like behavior. In order to avoid this elastic behavior against boundaries, a coefficient of restitution force ($\vec{f}^{\,cor}$) was taken into account when computing the normal component of boundary forces (lines 9 and 17 in Alg. 2). The proposed method is based on Caserta et al. [JCNCG16] which present a mass-spring-damper influenced by the research of the coefficient of restitution by Stevens and Hrenya [SH05], where $\vec{f}^{\,cor}$ is determined using Eq. 10.

$$\vec{v}_{bp} = \vec{v}_b - \vec{v}_p \tag{10a}$$

$$\vec{f}^{\,cor} = (\vec{v}_{bp} \cdot \vec{n})\vec{n}\Gamma \tag{10b}$$

Here $\vec{v}_{bp}$ is the relative velocity between the boundary particle, b, and the snow particle p, $\vec{n}$ is the normal of the collision, Γ is the user-defined restitution coefficient, and $\vec{f}^{\,cor}$ is the resulting force from the restitution coefficient in the normal direction.

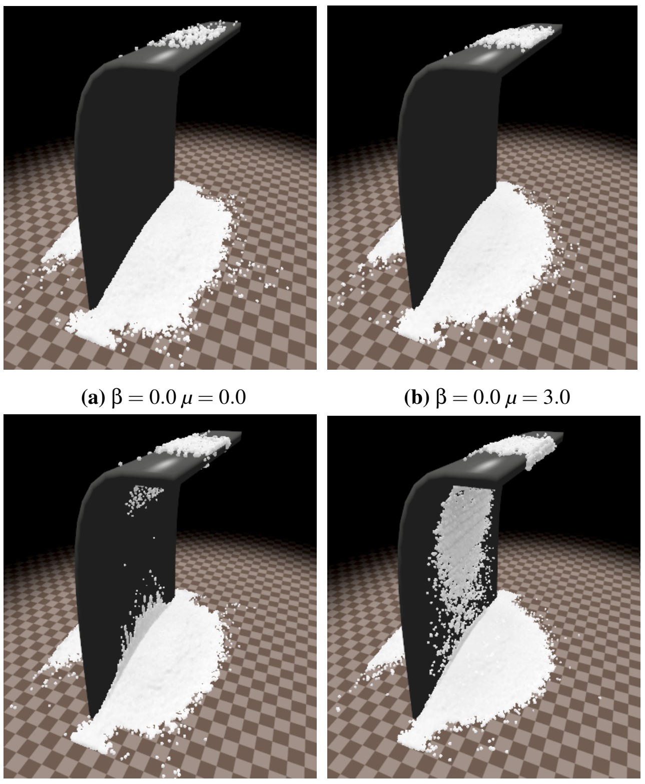

(a) β = 0.0 μ = 0.0 (b) β = 0.0 μ = 3.0 (c) β = 3.0 μ = 0.0 (d) β = 3.0 μ = 3.0

Figure 2: *A dry snowball (58K particles, ∆t = 0.6 ms) thrown at a boundary wall with different adhesion factors β and friction factors μ.*

In order to achieve a two-way coupling of snow with the rigid bodies, an accumulation of equal and opposite forces from the interacting snow particles is gathered for each solid body, which is then used to update its velocity and position. Tab. 1 summarizes all the forces and factors applied in our implementation and the nature of the particle interaction (snow-snow or snow-boundary) when they are applicable. This choice of force application is crucial to the working and stability of our solver. It is noteworthy that neither the cohesive normal force nor the normal force factor is used on the interaction of a snow particle with a static boundary particle. The repulsive component of the normal force though is still activated. Whereas the tangential force acts between interacting snow particles, the frictional force acts as the counterpart on the interaction of a snow particle with any boundary particle. Similarly, the boundary cohesion force replaces the usual normal cohesive force when a snow particle interacts with a boundary particle. Fig. 2 shows the behavior of the dry snow particles when thrown at a boundary wall for different adhesion factors β and friction factors μ.

Type	Snow-Snow	Snow-Boundary
Repulsive Normal Force	✓	✓
Cohesive Normal Force	✓	✗
Tangential Force	✓	✗
Normal Force Factor	✓	✗
Bond Threshold	✓	✗
Boundary Adhesion Force	✗	✓
Frictional Force	✗	✓
Coefficient of Restitution	✗	✓

Table 1: *Summary of various forces and factors applied when dealing with snow-snow or snow-boundary particle interactions in our solver.*

4.4. Time step

The smoothing length in SPH simulation is shown to have a significant influence in deciding the global time step [Mon05]. Iterative density solvers can step over this limitation and are able to employ large global time steps even while using the standard smoothing length. Unlike SPH, our smoothing radius is limited to the touching neighbors of any particle, which is roughly half of the standard smoothing length. Nonetheless, our solver is still able to use a time step as large as 1 ms even with large particle counts, which is a bit lower than [GHB*20]. At the same time, we significantly reduce the computational overhead incurred in every frame by reducing the number of neighbors and other expensive kernel computations. This allows us to sustain high frame rates and real-time to interactive behavior for a relatively large number of snow particles. Furthermore, we are able to employ time steps that are several times larger in magnitude than Hagenmüller et al. [HCN15].

Oscillation control: The particles could exhibit a vibrating, oscillatory behavior at larger time steps with the iterative implementation. This behavior is caused by particles that are maximally compressed but are surrounded by other particles. The already compressed particles can no longer reduce the radius and, therefore, use the accumulated force to move apart instead. However, the proximity of the particles makes them continuously collide with each other, moving back and forth. In order to prevent this, we incorporated an oscillation control inspired by the particle sleeping by Macklin et al. [MMCK14] was added (line 29 in Alg. 2). The oscillation control ensures that the positional integration only occurs if the particle has translated a distance above a certain threshold. The difference from particle sleeping is that our method uses a radius percentage as a threshold instead of a user-defined value.

5. Results

The proposed method was written using C++, GLSL, and NVIDIA CUDA. It was implemented and tested on a Windows machine with an Intel Core i5 9500F CPU clocked at 2.90GHz and an Nvidia RTX 2060 with 6GB VRAM. All presented images in the paper and the video were captured at a screen resolution of 1920x1080 pixels. We have rendered the particle surfaces in our scene using the Nvidia FLEX framework. Tab. 2 lists all the parameters tuned empirically corresponding to wet/dry snow and ice that we have used in our simulation. r_s is set to be in the range of 0.01 m - 0.04 m in the tested scenes.

Parameter	Name	Value	Unit
ϕ_{ds}	Bond Threshold Dry Snow	65	%
ϕ_{ws}	Bond Threshold Wet Snow	30	%
ϕ_i	Bond Threshold Ice	55	%
σ_{ds}	Normal Cohesion Dry Snow	17.95	*Pa*
σ_{ws}	Normal Cohesion Wet Snow	45.95	*Pa*
σ_i	Normal Cohesion Ice	506	*Pa*
Y_{ds}	Young's modulus Dry Snow	14000	*Pa*
Y_{ws}	Young's modulus Wet Snow	12000	*Pa*
Y_i	Young's modulus Ice	35000	*Pa*

Table 2: *Values of the physical parameters for dry snow (ds), wet snow (ws) and, ice used in our solver (i).*

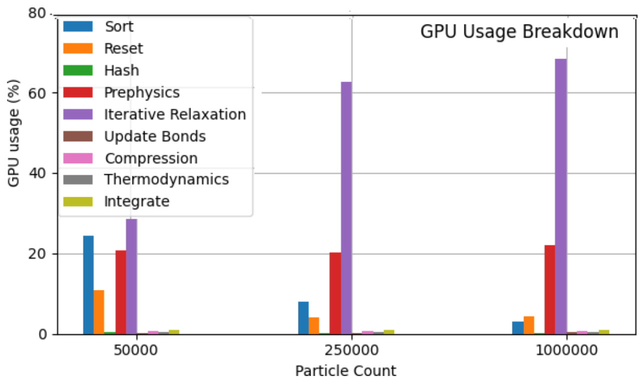


Figure 3: *Breakdown of the GPU usage when simulating for three seconds with wet snow and different particle counts at 1.0 ms time step.*

Fig. 3 gives a breakdown of the GPU usage of the various CUDA kernels involved in our implementation with increasing particle counts and a time step of 1 ms. The routines *Sort*, *Reset*, *Hash* are borrowed as such from the implementation of Green [Gre10], which are used to sort the particles based on their indices, reseting array values and computing hash index for particles, respectively. *Prephysics* includes the time spent on warm starting the iterative loop. It is clear that our method is able to achieve a good performance scalability with increasing particle counts.

5.1. States of snow

Our method can successfully simulate the visual behavior of dry and wet snow (Fig. 1,5,6), as well as a more hardened mixture tending to ice. This is made possible at efficient frame rates by our model with the help of cohesive forces and bonding. Snow in all the states exhibits a higher cohesion with the high frictional surfaces. At the same time, high frictional surfaces and hardened snow are more demanding computationally, as seen in the accompanying video. Our approach can, however, achieve interactive frame rates even with relatively large particle counts.

5.2. Comparison with the base method

Tab. 3 compares our iterative method with the base method in terms of timing efficiency for two different scenes. For each scene, par-

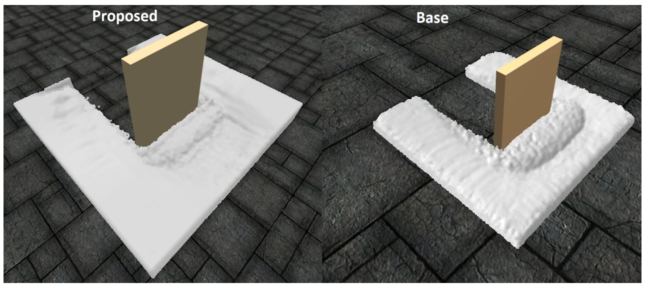
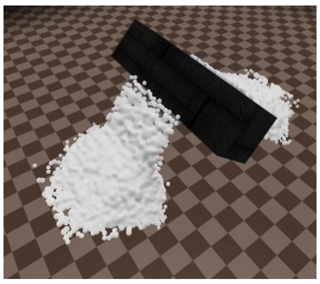
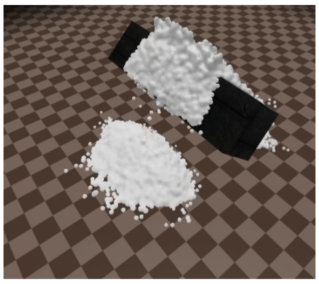


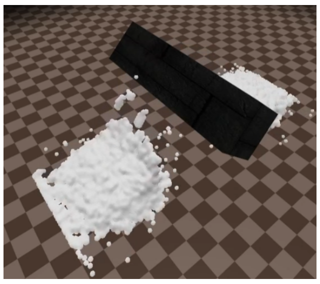
Figure 4: *Visual comparison of the proposed method running at* $\Delta t = 0.6$ *ms (left), and the base method [GMH19] at* $\Delta t = 0.1$ *ms (right), both consisting of 120K particles.*



(a) *Dry snow, Low friction*

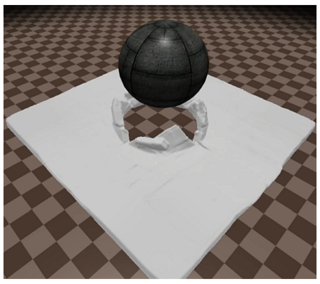


(b) *Dry snow, High friction*

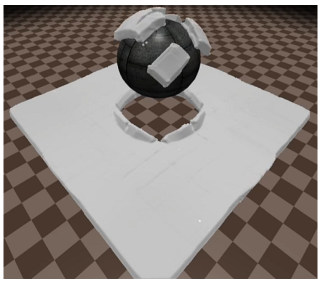


(c) *Wet snow, Low friction*

(d) *Wet snow, High friction*



(e) *Ice, Low friction*



(f) *Ice, High friction*

Figure 5: *Comparison of the behavior of dry, wet snow, and ice particles in our solver when dropped on a surface with low and high frictional properties (Δt = 0.6 ms, 120K particles).*

ticle counts are varied from 50K to 1M. Whereas the proposed method is timed with two different time steps Δt = 0.6 ms and 1.0 ms, the base method is running at the maximum possible Δt = 0.1 ms so that its stability is not compromised. Our approach achieves interactive frame rates even with around 500K snow particles. Furthermore, with one million snow particles, it runs nearly 8 times faster than the base method [GMH19]. The acceleration is visible even for lower particle counts. In Fig. 7, we visually compare the progression of our solver with the base method using two different time steps. Whereas both the methods provide a somewhat similar visual result towards the beginning of the compression, our approach can hold the snow mass for much longer, which is closer to the expected behavior.

Method	Particles	Δt (ms)	Time/Iter. (ms)	Exec. Time (s)	Speedup
Scene in Fig. 7					
Base	50k	0.1	2.09	62.73	-
Proposed	50k	0.6	7.14	35.66	1.76x
Proposed	50k	1.0	5.46	16.42	3.82x
Base	100k	0.1	3.55	106.20	-
Proposed	100k	0.6	11.49	57.26	1.86x
Proposed	100k	1.0	8.33	24.97	4.25x
Base	250k	0.1	8.70	260.40	-
Proposed	250k	0.6	20.83	102.74	2.53x
Proposed	250k	1.0	16.39	49.10	5.30x
Base	500k	0.1	19.23	573.70	-
Proposed	500k	0.6	35.71	175.87	3.26x
Proposed	500k	1.0	28.57	84.73	6.77x
Base	1000k	0.1	41.66	1245.35	-
Proposed	1000k	0.6	71.43	334.12	3.73x
Proposed	1000k	1.0	52.63	159.14	7.83x
Scene in Fig. 4					
Base	50k	0.1	2.35	63.09	-
Proposed	50k	0.6	7.94	28.05	2.25x
Proposed	50k	1.0	7.91	16.77	3.76x
Base	100k	0.1	4.34	108.02	-
Proposed	100k	0.6	11.91	39.95	2.7x
Proposed	100k	1.0	11.91	23.98	4.5x
Base	250k	0.1	11.49	269.53	-
Proposed	250k	0.6	25.32	80.57	3.35x
Proposed	250k	1.0	25.58	48.84	5.52x
Base	500k	0.1	28.35	601.41	-
Proposed	500k	0.6	50.86	147.68	4.07x
Proposed	500k	1.0	50.69	88.31	6.81x
Base	1000k	0.1	76.78	1469.89	-
Proposed	1000k	0.6	105.89	283.96	5.18x
Proposed	1000k	1.0	104.83	169.79	8.66x

Table 3: *Benchmark data are comparing the base [GMH19] and the proposed method running for three seconds of the simulation time with varying particle amounts, time-steps, and the respective speedup of the proposed method on each setting.*

5.3. Overlap analysis

Overlaps between the particles represent the error in our solver. The behavior of the system is directly influenced by the overlap evolution between neighboring particles over time. To this end, the overlaps were recorded and analyzed during three seconds of simulation time while a boundary object plowed through a thin sheet of particles, see Fig. 8. The dry and wet snow particles have significantly lower average overlaps (Fig. 8a), compared to fully compressed ice particles (Fig. 8b). The overlap linearly increases for dry and wet snow until the plowing is stopped. The ice particles can be seen as having more significant overlaps with less stable averages. After the boundary object has stopped, there is a period of finding the equilibrium state where the particles converge towards a stable average overlap different in all three tests. It is important to observe that our solver works more efficiently in the parameter range of dry and wet snow (Fig. 8a). Furthermore, our tests confirm that our particle overlaps are on an average less than in the base method [GMH19] while still operating at a ten times larger time step. However, the

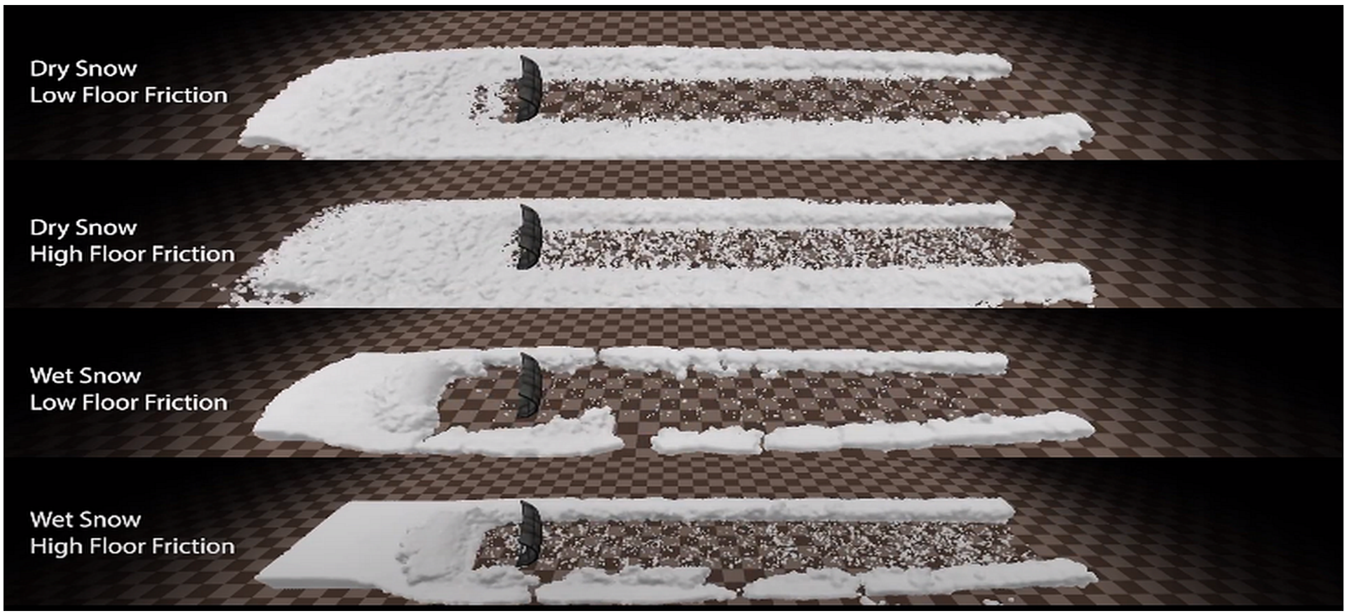


Figure 6: *Sheets of different snow types (consisting of 83K particles) and friction factors being plowed by a boundary object.*

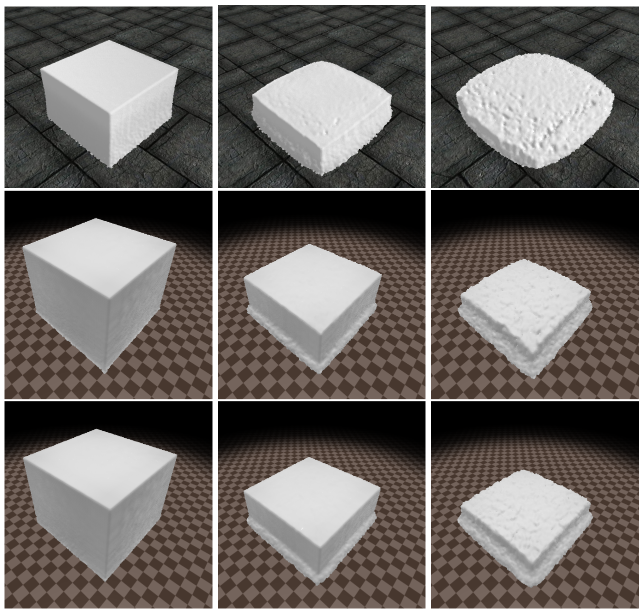
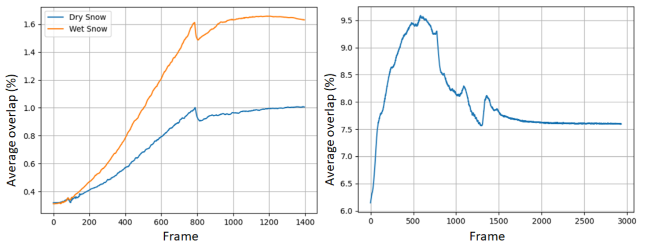


Figure 7: *Side-by-side comparison of the base method [GMH19] (upper row) and proposed method using wet snow configuration and $\Delta t = 0.6$ ms (middle row), $\Delta t = 1.0$ ms (bottom row) with 250k particles experiencing self compression. The rightmost images are taken after three seconds of simulation time.*

compressed snow is challenging as it begins to act as a rigid body mass and would, therefore, need additional computations to handle this more effectively. Fig. 9 shows the behavior of wet snow as it gets compressed when pushed between the pillars of a building.



(a) *Snow* **(b)** *Ice*

Figure 8: *Average particle overlaps comparison of dry and wet snow (left), and ice (right) being ploughed by boundary particles at $\Delta t = 1.0$ ms shown in Fig. 6.*

5.4. User study

In order to gauge the visual quality of snow obtained using our simulator for games, a user study was conducted wherein a qualitative questionnaire was distributed to 31 participants. Out of the 31 participants, 26 were male, four were female, and one preferred not to state the gender. The age span was between 20 and 28 years old, and 27 of these participants had experience in either playing or developing games. The participants were shown videos from our simulation (included in the accompanying movie submission) and were asked to pick up one of the ratings as shown in *Opinion* column of Tab. 4. The mapped numbers were not shown to the participants during the questionnaire. For the sake of a fair comparison, the questions showed a video consisting of different types of snow under the same set-up.

For an overview of the responses for the different types of snow, see Tab. 5. For all of the questions, the average and the median were calculated. The dry snow responses had the lowest average, with 3.37, with the most common response options being two and four. The wet snow had a higher average of 3.89 and the most com-

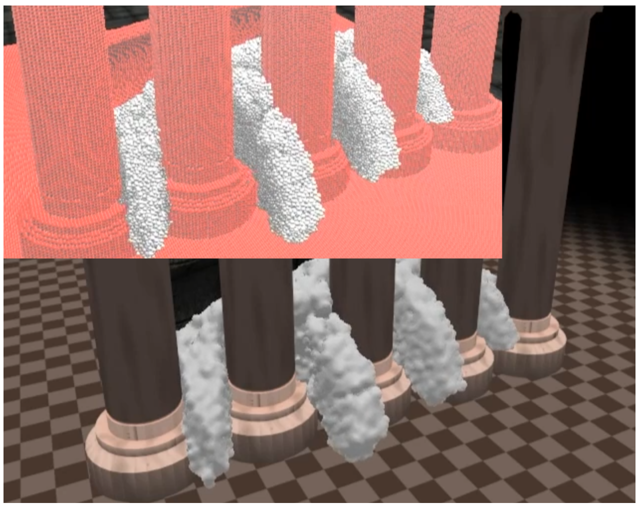


Figure 9: *Compression behavior of the wet snow (250K particles, Δt = 1.0 ms) in our solver when pushed between the boundary pillars, where darker particles represent a higher compression.*

mon response option of five. For both the ice and two-way coupling interaction, the most common answer was five, with an average of 4.15 and 3.88, respectively. The median of the responses was 4 for dry snow, wet snow, and the two-way coupling interaction. The ice configuration video had the highest possible median of 5. Overall, the questionnaire's values show that the participants found visual results from the proposed method more natural than unnatural.

In order to further determine whether this result applies to the general population, a statistical analysis was conducted. Since the questionnaire results do not match a normal distribution curve significance test was conducted using the *Mann-Whitney U-test* (also called *Wilcoxon rank-sum test*) [GC11]. To this end, the data was first divided into two different sets or partitions, as shown in the column *Set* of Tab. 4. The partitioning creates two comparable sets of data where a higher value indicates a stronger preference to that particular set. Following the table, the responses that found the proposed method natural were put in the "Natural" set, while responses that found the method to be unnatural were put in the "Unnatural" set. Responses that could not judge if the shown video was natural or unnatural were excluded from the analysis.

Response Option (RO)	Opinion	Value	Set
5	Natural	2	Natural
4	Slightly natural	1	Natural
3	Neutral	-	-
2	Slightly unnatural	1	Unnatural
1	Unnatural	2	Unnatural

Table 4: *Partitioning of the data from the questionnaire in either set "Natural" or set "Unnatural" depending on the option chosen by the participants. The Value after the partitioning represents by which amount the option was favored, where a higher value indicates either natural or unnatural behavior.*

To test for the statistical significance of the users' perception for favoring our method or not, a null hypothesis and research hypothesis was formulated as follows:

- ***H0** (null hypothesis)*: The appearance of the proposed method is perceived as unnatural at the 1% significance level.
- ***H1** (research hypothesis)*: The appearance of the proposed method is perceived as natural at the 1% significance level.

The statistical significance was determined using *Python* and the library *SciPy* [V*20]. The function `ranksums` was used with the two sets mentioned above. The resultant p-value of the function was 0.0047 at a 1% significance level. This result indicates that there is significant evidence to reject the null hypothesis. This further validates our assumption that the approximations introduced in our approach do not affect the visual value of our method for applications such as video games.

Scenario	RO 1	RO 2	RO 3	RO 4	RO 5
Dry snow	0	11	0	11	5
Wet snow	2	4	0	10	11
Ice/snow mixture	0	5	0	8	14
Density interaction	3	2	3	6	13

Table 5: *Number of votes per response option (RO) in the questionnaire as indicated in Tab. 4. Density interaction refers to the interaction of the solid objects of varying densities dropped on the mass of snow.*

6. Conclusions

We have presented an iterative DEM solution for efficient snow simulation, that can operate at real-time to interactive frame rates. Our particle-based method can handle large enough time steps for real-time performance and runs at high frame rates. Furthermore, the snow states can be varied by capturing the variation of density and bonding without penalizing significantly on the efficiency front. Our method easily enables the interaction of snow with complex boundary surfaces without losing efficiency. We have demonstrated that our approach can achieve a speed-up of nearly 8 times compared to the only existing real-time snow simulator for large particle counts. The visual realism value of our method is also validated with the help of a user study, wherein a majority of the participants voted in favor of the possible use of our solver in computer games and other such applications. The proposed solver also demonstrates that DEM-based solutions could be attractive to the computer graphics and gaming community, where computational efficiency is crucial.

In the future, the physical behavior of the compressed snow/ice particles could be improved by a more advanced formulation to handle them as a solid object. The neighborhood search operation could be optimized by maintaining multiple grids corresponding to different particle sizes. Currently, the high particle velocities resulting from the free fall under gravity limits the total movement of particles in a single frame, which a more sophisticated mechanism could resolve.

References

[AAT13] Akinci N., Akinci G., Teschner M.: Versatile surface tension and adhesion for SPH fluids. *ACM Transactions on Graphics 32*, 6 (Nov. 2013), 1–8. 6

[AO11] ALDUÁN I., OTADUY M. A.: SPH granular flow with friction and cohesion. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2011), SCA '11, Association for Computing Machinery, p. 25–32. 2, 4

[BK17] BENDER J., KOSCHIER D.: Divergence-free SPH for incompressible and viscous fluids. 1193–1206. Conference Name: IEEE Transactions on Visualization and Computer Graphics. 2, 4

[CEG*18] CORDONNIER G., ECORMIER P., GALIN E., GAIN J., BENES B., CANI M.-P.: Interactive Generation of Time-evolving, Snow-Covered Landscapes with Avalanches. *Computer Graphics Forum 37*, 2 (2018), 497–509. 2

[DGP16] DAGENAIS F., GAGNON J., PAQUETTE E.: An efficient layered simulation workflow for snow imprints. *The Visual computer 32*, 6 (2016), 881–890. 2

[FG11] FESTENBERG N. V., GUMHOLD S.: Diffusion-based snow cover generation. In *Computer Graphics Forum* (2011), vol. 30, Wiley Online Library, pp. 1837–1849. 2

[FHHJ18] FANG Y., HU Y., HU S.-M., JIANG C.: A temporally adaptive material point method with regional time stepping. In *Computer graphics forum* (2018), vol. 37, Wiley Online Library, pp. 195–204. 2

[FZ12] FAN N., ZHANG N.: Real-time Simulation of Rain and Snow in Virtual Environment. In *2012 International Conference on Industrial Control and Electronics Engineering* (Aug. 2012), pp. 29–32. 2

[GC11] GIBBONS J. D., CHAKRABORTI S.: *Nonparametric Statistical Inference*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. 10

[GEF15] GOSWAMI P., ELIASSON A., FRANZÉN P.: Implicit Incompressible SPH on the GPU. In *Workshop on Virtual Reality Interaction and Physical Simulation* (2015), Jaillet F., Zara F., Zachmann G., (Eds.), The Eurographics Association. 2

[GHB*20] GISSLER C., HENNE A., BAND S., PEER A., TESCHNER M.: An implicit compressible SPH solver for snow simulation. *ACM Transactions on Graphics 39*, 4 (July 2020), 36–1. 1, 2, 4, 6, 7

[GMH19] GOSWAMI P., MARKOWICZ C., HASSAN A.: Real-time particle-based snow simulation on the GPU. In *EGPGV 2019* (2019), Eurographics-European Association for Computer Graphics. 2, 3, 4, 8, 9

[Gre10] GREEN S.: Particle Simulation using CUDA, 2010. 3, 7

[GSSP10] GOSWAMI P., SCHLEGEL P., SOLENTHALER B., PAJAROLA R.: Interactive SPH Simulation and Rendering on the GPU. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Goslar, DEU, 2010), SCA '10, Eurographics Association, p. 55–64. 2

[HCN15] HAGENMULLER P., CHAMBON G., NAAIM M.: Microstructure-based modeling of snow mechanics: a discrete element approach. *The Cryosphere 9*, 5 (Oct. 2015), 1969–1982. 3, 4, 7

[HM09] HINKS T., MUSETH K.: Wind-driven snow buildup using a level set approach. In *Eurographics Ireland Workshop Series* (2009), vol. 9, pp. 19–26. 2

[ICS*14] IHMSEN M., CORNELIS J., SOLENTHALER B., HORVATH C., TESCHNER M.: Implicit Incompressible SPH. *IEEE Transactions on Visualization and Computer Graphics 20*, 3 (2014), 426–435. 4

[IUDN10] IWASAKI K., UCHIDA H., DOBASHI Y., NISHITA T.: Fast Particle-based Visual Simulation of Ice Melting. *Computer Graphics Forum* (2010). 4

[IWT12] IHMSEN M., WAHL A., TESCHNER M.: High-Resolution Simulation of Granular Material with SPH. In *Workshop on Virtual Reality Interaction and Physical Simulation* (2012), Bender J., Kuijper A., Fellner D. W., Guerin E., (Eds.), The Eurographics Association. 2

[JCNCG16] JORDAM CASERTA A., NAVARRO H. A., CABEZAS-GÓMEZ L.: Damping coefficient and contact duration relations for continuous nonlinear spring-dashpot contact model in DEM. *Powder Technology 302* (Nov. 2016), 462–479. 6

[JP20] JUNKER A., PALAMAS G.: Real-time interactive snow simulation using compute shaders in digital environments. In *International Conference on the Foundations of Digital Games* (2020), pp. 1–4. 2

[MM13] MACKLIN M., MÜLLER M.: Position based fluids. *ACM Trans. Graph. 32*, 4 (July 2013). 4

[MMC*20] MÜLLER M., MACKLIN M., CHENTANEZ N., JESCHKE S., KIM T.-Y.: *Detailed Rigid Body Simulation with Extended Position Based Dynamics*. Eurographics Association, Goslar, DEU, 2020. 2

[MMCK14] MACKLIN M., MÜLLER M., CHENTANEZ N., KIM T.-Y.: Unified particle physics for real-time applications. *ACM Transactions on Graphics 33*, 4 (July 2014), 1–12. 2, 7

[Mon05] MONAGHAN J. J.: Smoothed particle hydrodynamics. *Reports on Progress in Physics 68*, 8 (jul 2005), 1703–1759. 7

[MWXC04] MAO K., WANG M. Y., XU Z., CHEN T.: DEM simulation of particle damping. *Powder Technology 142*, 2-3 (2004), 154–165. 2

[NVI14] NVIDIA: GameWorks PhysX Overview, Feb. 2014. Accessed: 2021-04-28. 2

[RLD15] REYNOLDS D. T., LAYCOCK S. D., DAY A.: Real-time accumulation of occlusion-based snow. *The Visual Computer 31*, 5 (2015), 689–700. 2

[SGGP*20] SANCHEZ-GONZALEZ A., GODWIN J., PFAFF T., YING R., LESKOVEC J., BATTAGLIA P. W.: Learning to simulate complex physics with graph networks. 119. 3

[SH05] STEVENS A., HRENYA C.: Comparison of soft-sphere models to measurements of collision properties during normal impacts. *Powder Technology 154*, 2-3 (July 2005), 99–109. 6

[SP09] SOLENTHALER B., PAJAROLA R.: Predictive-corrective incompressible SPH. *ACM Transactions on Graphics 28*, 3 (July 2009), 1–6. 2, 4

[SSC*13] STOMAKHIN A., SCHROEDER C., CHAI L., TERAN J., SELLE A.: A material point method for snow simulation. *ACM Transactions on Graphics 32*, 4 (July 2013), 1–10. 1, 2, 5

[TF12] TAKAHASHI T., FUJISHIRO I.: Particle-based simulation of snow trampling taking sintering effect into account. In *ACM SIGGRAPH 2012 Posters*. 2012, pp. 1–1. 2

[TFN14] TAKAHASHI T., FUJISHIRO I., NISHITA T.: Visual Simulation of Compressible Snow with Friction and Cohesion. In *NICOGRAPH International* (2014). 3

[TKC21] TUMANOV E., KOROBCHENKO D., CHENTANEZ N.: Data-driven particle-based liquid simulation with deep learning utilizing sub-pixel convolution. *Proc. ACM Comput. Graph. Interact. Tech. 4*, 1 (Apr. 2021). 3

[TZWZ09] TAN Y., ZHANG X., WANG C., ZHAO Q.: Real-Time Snowing Simulation Based on Particle Systems. In *2009 First International Workshop on Education Technology and Computer Science* (Mar. 2009), vol. 3, pp. 7–11. 2

[V*20] VIRTANEN P., ET AL.: SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods 17* (2020), 261–272. 10

[WF15] WONG S.-K., FU I.-T.: Hybrid-based snow simulation and snow rendering with shell textures. *Computer Animation and Virtual Worlds 26*, 3-4 (2015), 413–421. 2

[WQS*20] WANG X., QIU Y., SLATTERY S. R., FANG Y., LI M., ZHU S.-C., ZHU Y., TANG M., MANOCHA D., JIANG C.: A massively parallel and scalable multi-GPU material point method. *ACM Transactions on Graphics (TOG) 39*, 4 (2020), 30–1. 2

[WR27] WHITEHEAD A. N., RUSSELL B. A. W.: *Principia mathematica; 2nd ed.* Cambridge Univ. Press, Cambridge, 1927. 3

[YLJ11] YINGCHAO ZHANG, LIPING ZOU, JIA LIU: Simulation of snow effects in visual simulation of virtual campus based on OSG. In *2011 International Conference on Multimedia Technology* (July 2011), pp. 3658–3662. 2