Eurographics Symposium on Parallel Graphics and Visualization (2019)
H. Childs, S. Frey (Editors)

# Real-time particle-based snow simulation on the GPU

Prashant Goswami[†] , Christian Markowicz[‡] & Ali Hassan[‡]

Blekinge Institute of Technology, Sweden

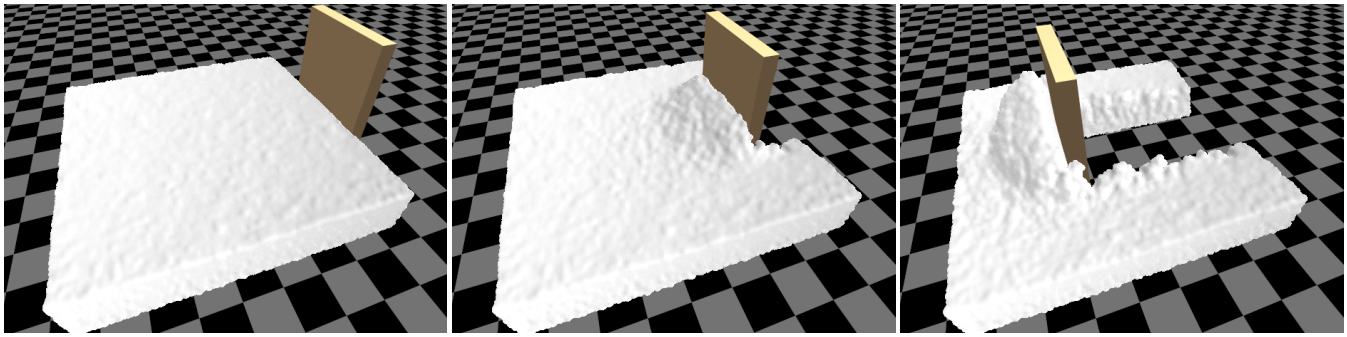


**Figure 1:** *A sliding wall plowing through the accumulated snow (consisting of 75.6K particles) on the ground simulated using our method, supporting both physics and visualization at around 282 frames per second.*

**Abstract**

*This paper presents a novel real-time particle-based method for simulating snow on the GPU. Our method captures compression and bonding between snow particles, and incorporates the thermodynamics to model the realistic behavior of snow. The presented technique is computationally inexpensive, and is capable of supporting rendering in addition to physics simulation at high frame rates. The method is completely parallel and is implemented using CUDA. High efficiency and its simplicity makes our method an ideal candidate for integration in existing game SDK frameworks.*

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Animation—Physical simulation

## 1. Introduction

Snow can enhance the realistic visual experience in computer games and other graphics applications. Snow is a cohesive material that can be compressed and altered over time into forms of ice and water. This produces a complexity to simulating the material as many factors, such as density and temperature, have an impact on how the material behaves. The existing methods in computer graphics often deal with static or accumulated snow that does not need to be animated [FO02]. The presence of snow in such scenes is usually captured by using textures or procedural noise, especially where the snow does not interact with other objects.

Efficient real-time methods exist for simulating various materials like fluid [MM13, GSSP10, GEF15], rigid and deformable bodies [MMCK14, NVI16]. GPU-based simulation method involving melting of ice to water droplets is given in [IUDN10]. Impressive offline technique computing snow dynamics using hybrid particle-Eulerian has recently been proposed in [SSC*13]. However, most of the existing methods capturing snow physics are computation intensive and hence offline, making them unsuitable for real-time applications. To the best of the knowledge of the authors, there is no published work on real-time snow simulation that besides modelling the realistic behavior of snow, can also be employed as a computionally viable and visually acceptable component in the games. Given the fact that soft snow is compressible, a simplified snow simulation can avoid the use of complicated incompressible solvers unlike those employed in fluid simulation. The efficiency part is even more important if snow simulation is not the primary focus of the scene, wherein it should take only an a reasonably acceptable fraction of the total frame time. In this paper, we introduce a parallel, purely particle-based technique to simulate snow physics

† prashant.goswami@bth.se
‡ the authors have equal contribution



DOI: 10.2312/pgv.20191109

in real-time on the GPU. The contributions of this paper are as following:

- A real-time, computationally inexpensive particle-based method to simulate snow dynamics on the GPU
- Capturing of cohesive bond-like behavior between snow particles
- Capturing of non-recoverable compression in the particle framework
- Incorporation of thermodynamics to capture phase change to water and ice in the particle framework

One major advantage of our proposed method besides its efficiency and simplicity, is that it can be easily incorporated in the existing unified particle-based frameworks.

## 2. Related Work

Procedural methods are extensively used in computer graphics to model snow. Snow simulation using grid based accumulation maps is one of the approaches used to simulate deposited snow. In [RLD15], snow accumulation is simulated in real-time by using surface-bound accumulation buffers mapped to objects in the scene. Through a shadow mapping technique snow fall is accumulated to buffers that are incremented. In turn, the accumulation displaces mesh-geometry. With a similar approach [Tok06] accumulates object-bound maps which are converted into polygon meshes. Accumulation maps are also used by [CZ13], where vehicle tracks in snow are simulated in real-time through terramechanics and terrain height displacements. The displacement is calculated by using the amount of dust particles generated from interaction. In [Fea00], snow accumulation is calculated through shooting particles upwards towards the sky from their accumulation location. A stability model is applied to cause avalanches at unstable areas.

In [ZCL10], the process of falling snow and snow accumulation is realized through a real-time particle simulation and displacement map. Values of the displacement map are accumulated by custom rules as particles collide with the ground, which in turn change the geometry. With a similar approach of interaction between particles and displacement map, [WWXP06], presents a real-time simulation including a three-dimensional wind field. The falling of snow is simulated in combination with deposition and erosion. Large scenes with snow are procedurally modeled by [GPG*16]. The model uses occlusion and temperature as parameters to offset vertices on an objects surface.

In [MP06], a real-time pressure driven wind system with particles for snow is presented. The approach uses a pressure gradient to calculate the wind speed of the particles. The gradient is calculated from low pressure zones below terrain peaks and high respective low pressure points within the world. The wind phenomenon is also taken into account by [MT10], where a pre-computed radiance transfer technique is used to calculate the accumulation of snow subjected to wind in real-time. The method is divided into a precompute step of wind vectors followed by an accumulation step. Snowflake movement with wind fields and accumulation of snow has also been modeled by [MMAL05]. [FO02] present a method to model snow drifts created by wind near obstacles. [FG11] propose a method to generate snow covers on complex scene geometries.

Previous research shows that the behavior of snow is complex; it depends on multiple factors such as density, humidity, and time [SSC*13]. The previous real-time research approaches have either simulated snow by two dimensional grids, limiting dynamic interactions in three dimensions, or by not incorporating important aspects, such as behavior of non-recoverable compression and phase transition. To widen the scope of what games current engines can produce, an approach to simulating snow, with the mentioned behaviors, is proposed.

Methods capturing physical behavior of fluids [MCG03, SP09] and materials like sand [SOH99, ZB05, AO11] are gaining popularity in computer graphics. Recently, some offline physics-based models for snow simulation have been developed. A real-time avalanche simulation is proposed by [GÖ14], where a mass spring model is used to simulate compressibility and attraction between particles and terrain. The approach simulates snow at a larger scale and does not take non-recoverable deformation into account. The motion of avalanches has been simulated by [TYDN10], where a method that deals with mixed-motion avalanches has been proposed. This is done by dividing snow into different layers that can interact; the different layers being suspension, dense-flow and accumulated. A grid-based approach is used for the suspension layer, while a particle-based method is used for the dense-flow and accumulated.

A semi-implicit material point method for the simulation of deformation and behavior of snow, including wet and dense snow, is presented by [SSC*13]. The method approximates the snow-mass as a continuum and evades the modeling of each snow grain. A method called Smoothed particle hydrodynamics (SPH) is used by [TF12] to capture an approximation of the behavior of snow. The simulation includes bond creation between grains and thermal conduction. [TFN14] model the compression of accumulated snow by introducing the *durability* factor which accounts for the amount of air volume trapped within the snow mass. Their method is based on Fluid Implicit Particle (FLIP) which is a hybrid method, using both Lagrangian particles and Eulerian grids.

Unlike the previous methods, our technique relies on a simple, inexpensive particle simulation instead of SPH to simulate the physics of accumulated snow in real time. Our approach easily captures the non-recoverable compression and incorporates thermodynamic properties of snow within this particle framework. Furthermore owing to the flexibility of particle-based method, the technique is also capable of handling the phase transition between snow, ice and water and their mutual interaction.

The remainder of this paper is organized as following. We describe our snow simulation model together with cohesion, compression and thermodynamics in Sec. 3. Sec. 4 presents the results obtained using implementation of our method, followed by conclusions and future work in Sec. 5.

## 3. Method

Our method is purely particle-based and to this end, we discretize snow into particles of uniform radius $r$. In order to achieve density change and phase transformation, the radius of the particle is altered keeping its mass unchanged. This helps us to compute the par-

ticle density based on its radius without carrying out more expensive kernel summation operation over the neighbors. Soft snow usually has a density of around $\rho_s = 100kg/m^3$ whereas hard snow (including ice) lies in the density range of $\rho_i = 800kg/m^3 - 900kg/m^3$ [BBH*11]. Snow particles begin with a sparse density (larger radius) and loose the entrapped air during compression to achieve a denser state. Our method assigns the state to a particle based on its density between these two extremes. This is achieved by storing the proportion of snow $\eta$ and ice $(1-\eta)$ corresponding to each snow particle based on its radus where the radius $r_s$ implies complete snow and $r_i$ complete ice state. Water particles have a constant density of $\rho_w = 1000kg/m^3$ in the simulation. Our unified model can handle all these states of snow and its transition to other forms like ice and water.

We employ a model similar to [Gre10] for neighborhood computation wherein a virtual grid is established in the simulation domain. A hash value is computed for each particle which maps it to a unique cell in this grid. The neighbors of each particle are found by querying its current and the 26 adjacent cells for particles that fall within support radius. In our case, this support radius is simply composed of neighboring particles touching the particle in question.

The major processes governing snow formation and its transformation are cohesive forces, compression and thermodynamics, see also Algo. 1 for the overview of our approach. In the following, we describe these processes in detail as applied to our model. Hereafter the particles in simulation are grouped to one of the sets of $S$, $I$ and $W$, which refer to snow, ice and water respectively when used in the algorithms.

**Algorithm 1** Snow Simulation

```
1:  while (animating) do
2:    for all particle i do
3:      find neighborhoods N_i(t)
4:    for all particle i do
5:      compute CohesionForces_i(t)
6:      compute Thermodynamics_i(t)
7:      compute Compression_i(t)
8:    for all particle i do
9:      update velocity v(t + Δt)
10:     update particle position x(t + Δt)
```

### 3.1. Cohesion

Snow particles bind together with each other through different forces in different states of matter. High liquid contents cause the snow to be cohesionless and slushy, while low liquid snow is well bonded [Col97]. In our work, we have focussed on the cohesive forces between solid particles ranging in the state from snow to ice. The cohesion present between particles in a heterogenous snow mixture is captured using the following forces.

#### 3.1.1. Cohesive forces

We adopt our cohesion model from [HCN15]. Since their model deals with only hardened snow, we extend it for soft snow and a mixture consisting of all intermediate densities for our purpose. The normal force acting between two snow particles in contact with each other, which includes both elastic compression and cohesive attraction, is given by Eq. 1.

$$\vec{F_n} = \begin{cases} -(Er\delta)\vec{n} & \text{if } -Er\delta < 4\sigma_n r^2 \\ 0 \text{ and cohesion is broken} & \text{if } -Er\delta \geq 4\sigma_n r^2 \end{cases} \quad (1)$$

Here $\mathbf{n}$ is the normal direction connecting the particles, $\sigma_n$ is the cohesive strength in the normal direction, $\delta$ is the overlap between the two particles, $E$ is the Young's modulus and $r$ the radius of the particles in the system. Notable is the fact that this model uses spring-based force between the particles to simulate the elastic deformation according to Hooke's law with the spring coefficient $k = Er$. An interesting property of $\vec{F_n}$ is that the force between two particles in proximity is attractive when they do not overlap ($\delta \leq 0$) and repulsive otherwise ($\delta > 0$). Furthermore, the strength of this force is governed by the rigidity of snow; for solidified snow/ice larger $E$ creates stronger force than for the soft snow.

However, we need to modify this formulation for our purpose since we assume that the snow mixture is heterogenous consisting of snow, ice and in-between transition particles. Soon after the creation most snow particles exist at densities between $\rho_s$ and $\rho_i$ due to the effects of compression and temperature. This is accounted by interpolating particle attributes like $E$, $\sigma_n$ in proportion to its snow $\eta$ and ice content $(1-\eta)$. As different radii and Young's modulus are used for different particles, the equation is modified to account for Newton's third law of opposite and equal forces for a pair of interacting particles $j$ and $k$ to

$$\vec{F_n} = \begin{cases} -\frac{E_j r_j + E_k r_k}{2}\delta\vec{n} & \text{if } -\frac{E_j r_j + E_k r_k}{2}\delta < 4\frac{\sigma_{nj} r_j^2 + \sigma_{nk} r_k^2}{2} \\ 0 \text{ and cohesion is broken,} & \text{otherwise} \end{cases} \quad (2)$$

In addition to the normal force, the particles also experience shear friction with their overlapping neighbors which inhibits their tangential movement. The tangential contact force in [HCN15] is simplified to only use the frictional part to Eq. 3

$$\vec{F_t} = (\vec{u_t}/|\vec{u_t}|)|\vec{F_n}|\vec{tan}(\varphi) \quad (3)$$

which excludes the use of parameter $\sigma_t$. Here $\vec{u_t}$ is the accumulated shear displacement between two particles and $\varphi$ the friction angle. This modification in essence amounts to the assumption that no tangential bonds exist between the particles. This simplifies our computations to keeping only normal bonds between particles without having to account for tangential bonds. Note that these contact forces are applied only to snow and ice and not water particles. The complete procedure to compute cohesive forces is outlined in Algo. 2 and these forces are described in detail below. Additionally, all particles in touch with the ground experience a frictional force.

#### 3.1.2. Bond formation

Bond formation plays a crucial role in the snow physics. As is visible from Eq. 1, the cohesion or bond between a pair of neighboring particles is broken beyond the threshold force limit when

they are not overlapping. This implies that once the bond between a pair of particles is broken, it is not reestablished. Such particles therefore, do not exert cohesive influence on each other, even if they come within the force range of $-Er\delta < 4\sigma_n r^2$ thereafter. The capturing of this bond breaking phenomenon in snow is essential to realistically model its behavior, especially during impact with other objects. In a naive manner, this can be implemented by storing the information on existing and broken bonds between each pair of particles (bond creation time, bond strength etc.). However, this not only creates a much higher memory requirement to store all bond information but also entails fetching and iterating through the list of valid bonds in order to compute cohesion force. Our experiments confirmed a significant reduction in the efficiency while implementing the aforementioned bonding on the GPU.

In order to circumvent this limitation, we introduce a lightweight approximation to mimic the normal bonding behavior between snow particles. We store the maximum count of neighbors $n_{max}$ for each particle encountered so far and update it in each loop with the current figure $n_{curr}$. Any particle with ratio $\frac{n_{curr}}{n_{max}}$ less than a certain threshold (set to 0.75) is tagged to have broken bonds. This kind of sudden reduction in number of neighbors can happen during collisions or breaking of snow mass since these particles are otherwise bonded together and undergo only slow and gradual relative movements. Any two particles with broken bonds within contact range of each other do not enter into bond formation or cohesive influence. However, the repulsive and the tangential forces between these particles continue to act if they come within overlapping contact (line 5-6 in Algo. 2).

### 3.1.3. Weak forces

Weak attractive forces come into play between water-ice molecules. For the sake of simplicity and efficiency, the cohesive interaction is handled using interfacial tension forces model similar to [IUDN10], given by Eq. 4, where $k_w$, $k_s$ and $k_i$ are interfacial constants. If colliding with a snow or ice particle, the water particle is repelled by a force proportional to the overlaps $\delta$. However, a more sophisticated way could be used to handle these interactions. Interaction between water particles can be handled using SPH forces.

$$\vec{F}_{weak} = \sum_{w \in N^{water}} k_w \frac{\vec{x}_w - \vec{x}}{||\vec{x}_w - \vec{x}||^2} + \sum_{s \in N^{snow}} k_s \frac{\vec{x}_s - \vec{x}}{||\vec{x}_s - \vec{x}||^2} + \sum_{i \in N^{ice}} k_i \frac{\vec{x}_i - \vec{x}}{||\vec{x}_i - \vec{x}||^2} \quad (4)$$

The sum total of all obtained forces $\vec{F}$ ( $= \vec{F}_n + \vec{F}_t + \vec{F}_{weak} + \vec{F}_g$) is used to update the particle velocity and its position.

## 3.2. Compression

Snow compresses due to self weight and external forces applied on the snow mass. The interconnecting bonds in snow are relatively small compared to the grain size and thus initially most of the deformation occurs in the bonds and not in the grains. As those bonds fail, stress is transferred to the remaining bonds that are intact leading to plastic deformation and hence energy loss.

**Algorithm 2** *CohesionForces$_i(t)$*

1: **for all** neighbor particle $k$ of $i$ **do**
2: **if** ($i \in$ S $\cup$ I and $k \in$ S $\cup$ I) **then**
3: **if** ($i$ has no broken bonds) and ($k$ has no broken bonds) **then**
4: calculate cohesive and tangential contact forces using (Eq. 2, 3)
5: **else if** ($i$ and $k$ overlap) **then**
6: calculate repulsive and tangential contact forces using (Eq. 2, 3)
7: **if** ($i \in$ W) or ($k \in$ W) **then**
8: calculate interfacial tension forces (Eq. 4)
9: Determine if the particle $i$ has broken bonds

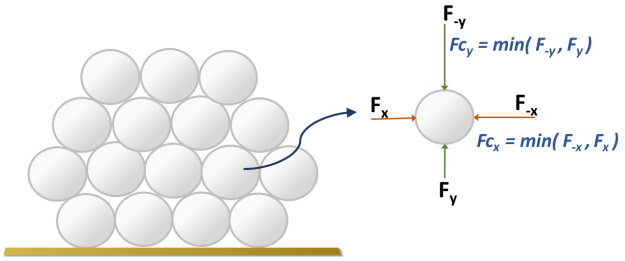



**Figure 2:** *Free body diagram for the compression on a snow/ice particle illustrated for 2D case. The net compressing force ($\vec{F}_c$) acting along each axis is the minimum of two forces $\vec{F}_{-x}$(or $\vec{F}_{-y}$) and $\vec{F}_x$(or $\vec{F}_y$) such that an equivalent non-zero force is opposing it.*

The compression itself is caused as a result of pair of opposing forces acting on a body. As illustrated for 2D counterpart in Fig. 2, only the force component receiving a non-zero opposite reaction contributes towards compressing a body. The balance force leads to other forms of movements like translation, rotation etc. To determine the compressive component $\vec{F}_c$, all acting non-cohesive forces on the particle are resolved along $\pm\mathbf{F}_x$, $\pm\mathbf{F}_y$ and $\pm\mathbf{F}_z$ axes. The magnitude of $\vec{F}_c$ is captured by Eq. 5.

$$F_c = \sqrt{min(\vec{F}_{-x}, \vec{F}_{+x})^2 + min(\vec{F}_{-y}, \vec{F}_{+y})^2 + min(\vec{F}_{-z}, \vec{F}_{+z})^2} \quad (5)$$

The compression function in Algo. 3 starts with calculating the compressive force $F_c$ with the help of Eq. 5. In the last step, the new radius is calculated by using the compressive force $F_c$. Similar to [TFN14], compression is implemented with a durability variable $d$ which is kept for every particle. The durability represents the proportion of air trapped within a snow particle which is highest ($d = 1$) when the snow particle is uncompressed. The term $d$ is linearly transformed to reduce the particle radius which in turn

changes the density of snowpack. The original formulation was modified by adding a conditional statement, see Eq. 6.

$$d \leftarrow \begin{cases} d - k_q p_c, & \text{if } \vec{F}_c > \vec{D}(\rho_i) \\ d & \text{otherwise} \end{cases} \tag{6}$$

where $p_c$ is the pressure generated from the compressive force $F_c$. $k_q$ denotes the durability change coefficient, defined as the radius change caused by unit pressure. The threshold function for plastic compression $D(\rho_i)$ is calculated by approximating the shape of the stress-density curve for snow from [BBH*11] as

$$\vec{D}(\rho_i) = \vec{F}_{minW} + (\frac{e^{\frac{\rho_i}{100}-1} - 0.000335}{2980.96})\vec{F}_{maxW} \tag{7}$$

where $F_{minW}$ is a constant of the minimum amount of force a particle can withstand prior to initial compression. It is defined as a value with a magnitude that is slightly larger than $mg$ (where $m$ is particle mass and $g$ gravitational constant).

**Algorithm 3** $Compression_i(t)$

1: calculate compressive force using Eq. 5
2: mirror forces on boundary particles for boundary reaction
3: calculate compressed radius using Eq. 6

### 3.3. Thermodynamics

In [IUDN10], the heat transfer between SPH particles for melting is formulated as

$$\frac{\Delta T_i}{\Delta t} = \alpha \sum_{j \in N_i} m_j \frac{(T_j - T_i)}{\rho_j} \nabla^2 W(x_{ij}, H_h) \tag{8}$$

where $\alpha$ is the thermal diffusion coefficient, $N_i$ a set of neighboring particles whose distances are smaller than $H_h$ or support radius from particle $i$, $W$ is the smoothing kernel and $x_{ij}$ is the distance between $x_i$ and $x_j$. The diffusion coefficient $\alpha = \frac{k_c}{\rho C_p}$ is constant with parameter $k_c$ set to thermal conductivity and $C_p$ to the specific heat capacity value. However, in our case in order to show the heat transfer between particles in real-time the thermal diffusion coefficient is scaled up by a factor $S_f$. The symbol $\Delta T_i$ is the obtained change in temperature for particle $i$ which is then employed to compute the heat loss/gain with the neighboring particles as

$$Q_{i_{neighbors}} = C_p m \Delta T_i \tag{9}$$

Heat exchange between the snow particle $i$ and the air, $Q_{i_{air}}$ is computed using Newton's law of cooling air

$$Q_{i_{air}} = h_T(T_{air} - T_i) \quad \delta A_{air} \tag{10}$$

where $h_T$ is the heat transfer coefficient, $T_{air}$ the temperature for air, $T_i$ the temperature for a particle $i$ and $\delta A_{air}$ is the area of particle $i$ exposed to the air. A similar expression is obtained for heat exchange between the particle and the ground, $Q_{i_{ground}}$ by replacing $\delta A_{air}$ with $\delta A_{ground}$ which is the exposed area of particle to ground. The area exposed to ground or air for each particle is determined by the expression $\delta A = \frac{6-n}{6} A$ as laid out in [IUDN10], where $n$ is the number of neighboring snow, ice and water particles and $A$ is the total surface area for the particle in consideration. $\delta A$ is clamped to 0 when $n$ exceeds 6. $h_T$ for a snow particle is interpolated using its snow content $\eta$ and $h_{T_{snow}}$ and $h_{T_{ice}}$.

As a particle does not melt directly as it reaches to $0^\circ$ $C$, the latent heat is stored in terms of amount of water melted within the particle using parameter $\beta$. The latent heat is calculated by modifying the standard equation into

$$Q_f = mL\beta \tag{11}$$

where $L$ is the specific latent heat of fusion for the material. The entire particle is converted into water as soon as $\beta$ reaches a value of 1.

The total heat value received for a particle is calculated by

$$Q_i = Q_{i_{air}} + Q_{i_{ground}} + Q_{i_{neighbors}} + Q_f \tag{12}$$

and is used to change its temperature. The temperature change computation is limited to the melting point of snow, after which further heat is used for phase transition.

### 3.4. Discussion

Our approach of computing particle density using its radius and not with kernel based summation over the neighbors directly comes from the observation that solid snow is neither fluid nor incompressible and hence employing SPH or incompressible solvers would be a computational overkill for snow simulation. Owing to this simplification, we do avoid expensive interpolation kernels in a separate CUDA kernel. In Eq. 8, $\nabla^2 W(x_{ij}, H_h)$ is calculated from a simple box kernel.

In order to capture the relative density between snow and ice and also to simplify our computations, we have set $r_s = r_w = 2 * r_i$ in our simulations. We continue to keep single grid for neighborhood computation for all particle sizes. However, the implementation could be made more efficient by keeping two separate grids for particles with size $r_s$ and $r_i$. The particle set is divided into chunks of 32/64 and a CUDA kernel is launched for each of these chunks. Each particle is assigned a CUDA thread for attribute computation and separate kernels are launched for force, thermodynamics and compression computation.

## 4. Results

The presented method was written in C++ and CUDA (version 9.1) and tested on a Windows PC with Intel(R) Xeon(R) 2.9 GHz quad core processor and Nvidia GeForce GTX 1080 graphics card. Since

**Figure 3:** *Sphere composed of fresh snow (4.2K particles) dropped from a height, (left) without broken bonds, (right) with our tracked bonding approximation between snow particles. Particles with broken bonds (colored brown) do not enter into cohesive interaction or bond formation with similar neighbors.*

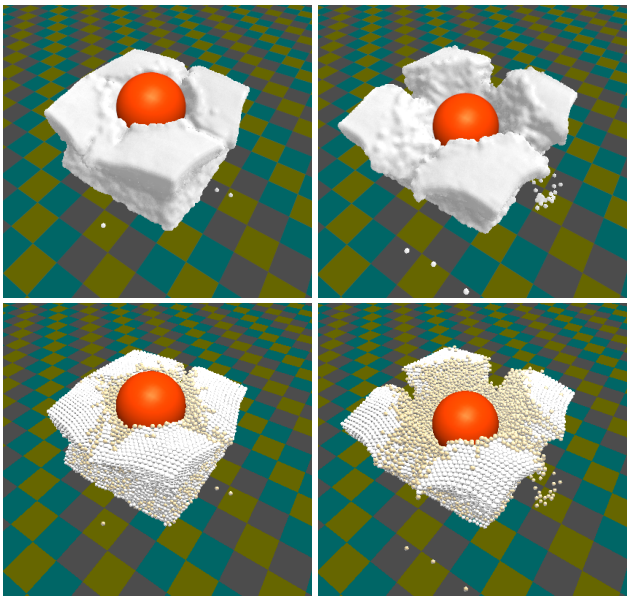


**Figure 4:** *Varying weights w dropped on a block of compressed snow (left)w = 2 kg and (right)w = 5 kg. Particle visualization for corresponding scenes is shown in the images below where non-white particles represent broken bonds.*

| Variable | Datatype |
|---|---|
| Position + Radius ($\vec{x}(t)$) | float4 |
| Velocity ($\vec{v}(t)$) | float4 |
| Neighbor count | uint16 |
| Phase ($\eta$) | float |
| Amount of Water ($\beta$) | float |
| Temperature (T) | float |

**Table 1:** *Attributes stored for each particle in our CUDA implementation.*

| Parameter | Meaning | Value |
|---|---|---|
| $r_s$ | Sparse snow radius | 0.05 $m$ |
| $r_i$ | Dense snow/ice radius | 0.025 $m$ |
| $E_{snow}$ | Young modulus snow 100 $kg/m^3$ | 5000 $Nm^{-2}$ |
| $E_{ice}$ | Young modulus ice 900 $kg/m^3$ | 35000 $Nm^{-2}$ |
| $\varphi$ | Angle of repose | 38° |
| $k_q$ | Durability coefficient | 0.000005 $m^3N^{-1}$ |
| $\sigma_{ns}$ | Normal cohesion strength snow | 625 $Nm^{-2}$ |
| $\sigma_{ni}$ | Normal cohesion strength ice | 3750 $Nm^{-2}$ |
| $F_{minW}$ | Min force a particle can withstand | 0.12275 $N$ |
| $F_{maxW}$ | Max force a particle can withstand | $10^4$ $N$ |
| $C_{p_{water}}$ | Specific heat capacity water | 4186 $Jkg^{-1}K^{-1}$ |
| $C_{p_{snow}}$ | Specific heat capacity snow | 2090 $Jkg^{-1}K^{-1}$ |
| $C_{p_{ice}}$ | Specific heat capacity ice | 2050 $Jkg^{-1}K^{-1}$ |
| $k_s/k_i/k_w$ | Interfacial tension coefficients | 0.00012 |
| $h_{T_{water}}$ | Thermal conductivity of water | 0.602 $Wm^{-2}K^{-1}$ |
| $h_{T_{snow}}$ | Thermal conductivity of snow | 0.1 $Wm^{-2}K^{-1}$ |
| $h_{T_{ice}}$ | Thermal conductivity of snow | 0.7 $Wm^{-2}K^{-1}$ |

**Table 2:** *Parameter setting in our simulation. Some values have been altered and experimentally tuned in order to allow for a reasonably large time step for real-time purposes. We verified through our experiments that these alterations did not have a significant impact on the nature of simulation.*

the primary focus of this work is to simulate snow efficiently, we have chosen to render the surface using Nvidia FLEX framework which is highly efficient. The quality of the rendered surface can be further improved by using dedicated rendering methods such as [NIDN97]. In order to keep the simulation progress in real-time, the actual Young's modulus for ice cannot be used as this would require a very small time-step which would slow down the simulation significantly. Stable values for the lowest density snow $E_s$ and that of ice $E_i$ were instead captured by testing. These values are interpolated for other densities throughout the simulation. Other density dependent attributes for particles, for example cohesive strength,

| Particle count | Scene | Rendering-**surface** | Rendering-**particles** |
|---|---|---|---|
| 8000 | Fig. 8 | 734 | 1023 |
| 20000 | Fig. 6 | 580 | 783 |
| 35937 | Fig. 4 | 430 | 607 |
| 75615 | Fig. 1 | 282 | 335 |
| 111000 | Fig. 5 | 230 | 268 |

**Table 3:** *Performance statistics: frame rates per second for the scenes with varying particle counts, both when rendered as surface and particles (sprites).*

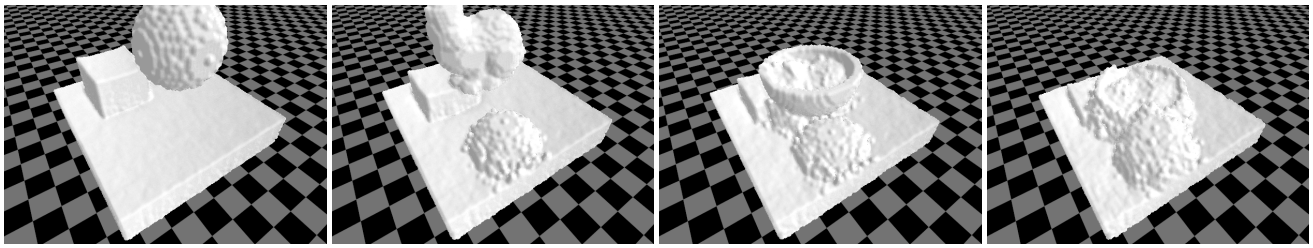



**Figure 5:** *Various snow shapes composed of fresh snow are dropped on an existing snow block leading to a total particle count of 111K.*

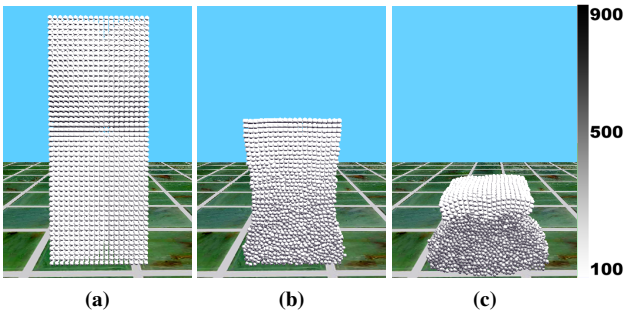



**Figure 6:** *Time-lapse visualization of the density of snow particles in a tall snow column consisting of 20K particles (initially fresh snow), gradually undergoing self-compression. The particles are colored in accordance to their density, with white representing the minimum (100) and black maximum (900) in kg* $m^{-3}$.

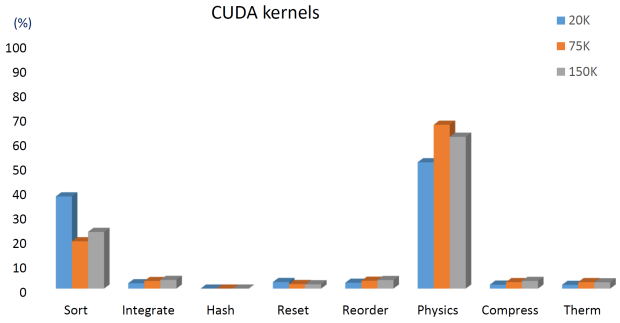



**Figure 7:** *Time spent in various CUDA kernels in our simulation for three different particle counts (rendering excluded). Base physics and sorting takes up most of the CUDA time while other kernels like those computing particle compression and thermodynamics have negligible impact.*

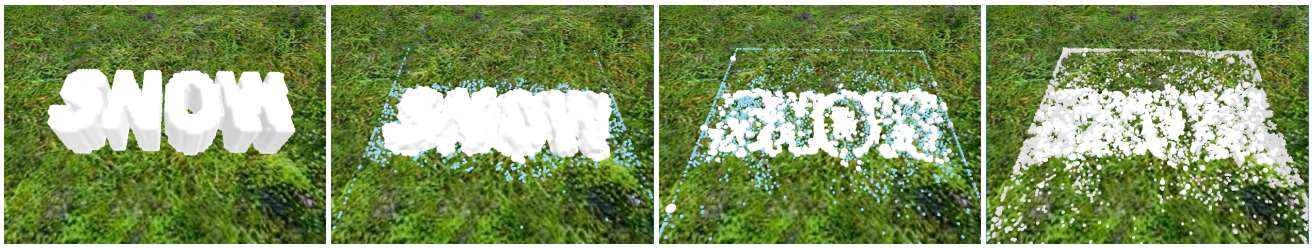
are similarly obtained by interpolating between $\sigma_{ns}$ and $\sigma_{ni}$ based on snow/ice content $\eta$. We used a time step $\Delta t = 10^{-3}$ for all the shown examples and the particle radius range from 0.025 to 0.05 meters. The parameter values used in our setting are as listed in Tab. 2. The attributes stored for each particle in our implementation are given in Tab. 1. Another advantage of our method is that no manual setting of unintuitive, complicated parameter values is required to produce the simulation.

Fig. 1 shows a scenario with 75.6K particles where accumulated snow is plowed with a plane. The simulation together with rendering is running at around 282 frame rates per second. In Fig. 3, a snow sphere consisting of 4169 particles is dropped on the ground and comparison is made between broken bonds using our method (right) vs. no broken bonds (left) between the particles. The interaction of different weights when dropped from the same height on a pile of somewhat solidified snow using our method is demonstrated in Fig. 4. Both these scenarios reinforce our assumption of the importance of capturing bond breaking phenomena for realistic snow simulation. Further, it is evident from Fig. 4 that a heavier weight breaks more bonds in the solidifed snow than the lighter one. Fig. 6 shows the time-lapse of compression of fresh snow due to self weight, where higher density is represented by darker color. The volume of the condensed snow reduces considerably as compared to its initial state owing to particle compression and state transformation thereof. The melting behavior of snow using our method which includes snow, ice and water particles is shown in Fig. 8. The ground and air temperature is first raised to 5°C that melts snow and then reduced to -1°C making the melted snow freeze to ice. The method runs at high frame rates when rendered with surface even for 111K particles, see also Fig.5 where different snow shapes are dropped on top of each other. Tab. 3 lists the performance statistics for our method for the shown scenes with increasing particle count.

Fig. 7 graphically represents the time spent on individual CUDA kernels as a percentage of total physics time using CUDA. For high particle counts, the rendering takes about 70% of the total frame time with the surface and 65% with the opengl sprites. Therefore, the physics computation part is relatively lightweight and can run at real-time rates even for higher particle counts. Of the total time spent on CUDA kernels, about 63% is on the physics kernel, 23% on sorting while compression and thermodynamics take less than 1% each. Furthermore, our bond tracking estimation consumes negligible physics time.

## 5. Conclusions

We have presented a particle-based, computationally inexpensive and yet convicingly realistic method for real-time snow simulation.



(a) (b) (c) (d)

**Figure 8:** *Fresh snow slowly melting at ground and air temperature of 5° C (a-c). Water refreezes to ice (in light grey color) in (d) when the air temperature is reduced to -1° C.*

The presented method is tailored for applications like games and provides a good trade-off between efficiency and realism. Our technique captures the bonding behavior between particles using a simple approximation and the non-recoverable compression of snow to ice. Furthermore, our framework can effectively incorporate the thermodynamics to capture phase change from snow to ice and water. The presented model is easy to implement and can be efficiently incorporated to simulate snow in computer games and virtual environments. To this end, the method can support very high frame rates even for relatively large particle counts, and hence can easily exist as one of the background components in any game.

For future direction, better interaction of snow with the material boundaries can be studied. Currently we model only dry snow. Our technique can be made richer by incorporating more parameters from [SSC*13] to enhance the wetness/dryness ratio of snow material. Another promising research direction could be to develop dedicated models to simulate hardened (incompressible) snow.

## References


[AO11] ALDUÁN I., OTADUY M. A.: Sph granular flow with friction and cohesion. In *Proc. of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (2011). doi:10.2312/SCA/SCA11/025-032. 2

[BBH*11] BISHOP M. P., BJÖRNSSON H., HAEBERLI W., OERLEMANS J., SHRODER J. F., TRANTER M., SINGH V. P., SINGH P., HARITASHYA U. K.: *Encyclopedia of snow, ice and glaciers*. Springer Science & Business Media, 2011. 3, 5

[Col97] COLBECK S. C.: *A Review of Sintering in Seasonal Snow.* Tech. rep., DTIC Document, 1997. 3

[CZ13] CHEN X., ZHU Y.: Real-time simulation of vehicle tracks on soft terrain. In *International Symposium on Visual Computing* (2013), Springer Berlin Heidelberg, pp. 437–447. 2

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

[FG11] FESTENBERG N. V., GUMHOLD S.: Diffusion-based snow cover generation. *Computer Graphics Forum 30*, 6 (2011), 1837–1849. doi:10.1111/j.1467-8659.2011.01904.x. 2

[FO02] FELDMAN B. E., O'BRIEN J. F.: Modeling the accumulation of wind-driven snow. In *ACM SIGGRAPH 2002 Conference Abstracts and Applications* (New York, NY, USA, 2002), SIGGRAPH '02, ACM, pp. 218–218. doi:10.1145/1242073.1242231. 1, 2

[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. doi:10.2312/vriphys.20151331. 1

[GÖ14] GÜÇER D., ÖZGÜÇ H. B.: Simulation of a flowing snow avalanche using molecular dynamic. *Turkish Journal of Electrical Engineering & Computer Sciences 22*, 6 (2014), 1596–1610. 2

[GPG*16] GROSBELLET F., PEYTAVIE A., GUÉRIN E., GALIN E., MÉRILLOU S., BENES B.: Environmental objects for authoring procedural scenes. *Comput. Graph. Forum 35*, 1 (Feb. 2016), 296–308. doi:10.1111/cgf.12726. 2

[Gre10] GREEN S.: Particle Simulation using CUDA. http://developer.download.nvidia.com/assets/cuda/files/particles.pdf, 2010. [Online; accessed 05-May-2017]. 3

[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* (Aire-la-Ville, Switzerland, Switzerland, 2010), SCA '10, Eurographics Association, pp. 55–64. 1

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

[IUDN10] IWASAKI K., UCHIDA H., DOBASHI Y., NISHITA T.: Fast particle-based visual simulation of ice melting. *Computer Graphics Forum 29*, 7 (2010), 2215–2223. 1, 4, 5

[MCG03] MÜLLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2003), SCA '03, Eurographics Association, pp. 154–159. 2

[MM13] MACKLIN M., MÜLLER M.: Position based fluids. *ACM Trans. Graph. 32*, 4 (July 2013), 104:1–104:12. doi:10.1145/2461912.2461984. 1

[MMAL05] MOESLUND T. B., MADSEN C. B., AAGAARD M., LERCHE D.: Modeling falling and accumulating snow. In *Institute of Mathematics and its Applications - Vision, Video and Graphics 2005, VVG 2005* (2005), pp. 61–68. 2

[MMCK14] MACKLIN M., MÜLLER M., CHENTANEZ N., KIM T.-Y.: Unified particle physics for real-time applications. *ACM Trans. Graph. 33*, 4 (July 2014), 153:1–153:12. doi:10.1145/2601097.2601152. 1

[MP06] MARTIN C., PARBERRY I.: Real time dynamic wind calculation for a pressure driven wind system. In *Proceedings of the 2006 ACM SIGGRAPH symposium on Videogames* (2006), ACM, pp. 151–154. 2

[MT10] MORIYA T., TAKAHASHI T.: A real time computer model for

wind-driven fallen snow. In *ACM SIGGRAPH ASIA 2010 Sketches* (2010), ACM, p. 26. 2

[NIDN97] NISHITA T., IWASAKI H., DOBASHI Y., NAKAMAE E.: A modeling and rendering method for snow by using metaballs. *Comput. Graph. Forum 16* (1997), 357–364. 6

[NVI16] NVIDIA: SDK. https://developer.nvidia.com/cuda-code-samples, 2016. [Online; accessed 05-May-2017]. 1

[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

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

[SP09] SOLENTHALER B., PAJAROLA R.: Predictive-corrective incompressible sph. *ACM Trans. Graph. 28*, 3 (July 2009), 40:1–40:6. doi:10.1145/1531326.1531346. 2

[SSC*13] STOMAKHIN A., SCHROEDER C., CHAI L., TERAN J., SELLE A.: A material point method for snow simulation. *ACM Trans. Graph. 32*, 4 (July 2013), 102:1–102:10. doi:10.1145/2461912.2461948. 1, 2, 8

[TF12] TAKAHASHI T., FUJISHIRO I.: Particle-based simulation of snow trampling taking sintering effect into account. In *ACM SIGGRAPH 2012 Posters, SIGGRAPH'12* (2012). 2

[TFN14] TAKAHASHI T., FUJISHIRO I., NISHITA T.: Visual simulation of compressible snow with friction and cohesion. 2, 4

[Tok06] TOKOI K.: A shadow buffer technique for simulating snow-covered shapes. In *Proceedings - Computer Graphics, Imaging and Visualisation: Techniques and Applications, CGIV'06* (2006), vol. 2006, pp. 310–316. 2

[TYDN10] TSUDA Y., YUE Y., DOBASHI Y., NISHITA T.: Visual simulation of mixed-motion avalanches with interactions between snow layers. *Visual Computer 26*, 6-8 (2010), 883–891. 2

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

[ZB05] ZHU Y., BRIDSON R.: Animating sand as a fluid. *ACM Trans. Graph. 24*, 3 (July 2005), 965–972. doi:10.1145/1073204.1073298. 2

[ZCL10] ZHANG J., CAI X., LI J.: Rendering snowing scene on gpu. In *Intelligent Computing and Intelligent Systems (ICIS), 2010 IEEE International Conference on* (2010), vol. 3, IEEE, pp. 199–202. 2