Snow Rendering and Simulation

https://github.com/DrJDen31/ACG_Final_Project

Tre'Vaughn Barboza
barbot@rpi.edu
Rensselaer Polytechnic Institute

Jaden Tompkins
tompkj3@rpi.edu
Rensselaer Polytechnic Institute

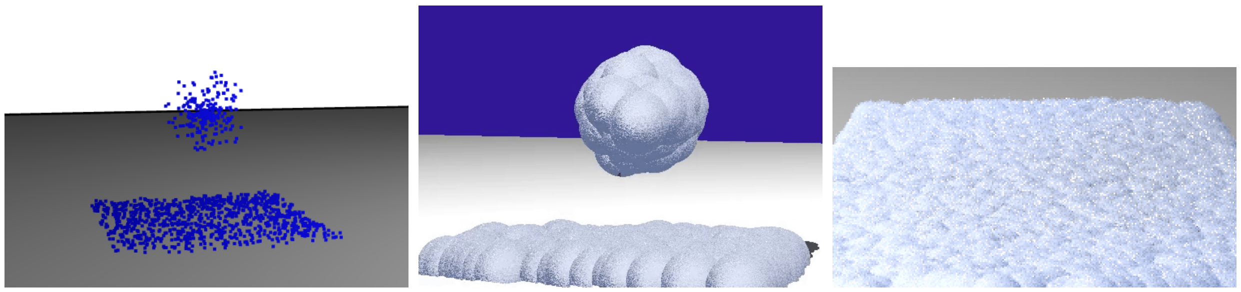

Figure 1: Demonstrations of snow simulation and rendering

ABSTRACT

We present a hybrid particle-grid simulation system for modeling snow, built on the Moving Least Squares Material Point Method (MLS-MPM). Our framework captures key physical behaviors of snow such as deformation, compaction, rolling accumulation, and avalanche-like dynamics by integrating elastoplastic material models with particle-to-grid transfer schemes. The system is tightly coupled with a rendering pipeline featuring bi-directional path tracing, enabling the visualization of characteristic snow effects including sparkle, volumetric scattering, and color attenuation. While the simulation operates offline due to the computational demands of accuracy and visual fidelity, the architecture supports modular expansion and interaction between physics and rendering, facilitating experimentation with complex snow phenomena in physically motivated scenes.

KEYWORDS

Snow simulation, MPM, MLS-MPM, Path tracing, Snow rendering, Particle-grid hybrid

1 INTRODUCTION

Snow presents a unique challenge in computer graphics due to its complex physical behavior and distinct optical characteristics. It can deform, fracture, clump and flow as granular material, while also exhibiting visually rich effects such as sparkles, translucency, and a bluish tint from subsurface scattering. Accurately capturing both its dynamic behavior and appearance requires an integrated approach.

We develop a unified framework for simulating and rendering realistic snow. Our approach is based on the Material Point Method (MPM), a hybrid particle-grid technique capable of modeling continuous material under large deformation. Building on this, we adopt the Moving Least Square (MLS-MPM) variable to improve the accuracy of interpolation and the simulation stability.

To capture the distinct appearance of snow, the system integrates a physically-motivated lighting model with the simulation. Key optical phenomena such as sparkle and bluish tinting are rendered using a custom ray tracing pipeline with support for bi-directional path tracing. This approach enables accurate modeling of subsurface scattering and light transmission within snow layers. The simulation and rendering components are tightly coupled, allowing for dynamic snow behaviors such as compaction, snow-to-snow interaction, and avalanche-like flows to be visualized with consistent physical and visual fidelity.

This integrated pipeline aims to support extensibility, previewing, and physically plausible results, laying the groundwork for future snow-related effects in both simulation and visuals.

2 RELATED WORK

This project builds on a foundation of work in physically-based simulation and snow rendering. The foundational technique used is the Material Point Method, a hybrid Lagrangian-Eulerian technique for simulating a variety of physical materials, originally introduced to extend FLIP methods to solid mechanics. The Disney Research paper, *A Material Point Method for Snow Simulation* (2013), demonstrated the effectiveness of MPM in simulating snow deformation, compaction, and fracture through a combination of particle-grid transfers and a specialized elastoplastic constitutive model.

Niall Lynch's practical guide to MPM (2022) provides a hands-on reference that demystifies both the original MPM framework and modern improvements focusing on MLS-MPM. This resource directly informed both implementation details and improvement strategies used during development of this project.

The Affine Particle-In-Cell (APIC) method improves energy conservation in grid-particle transfers and is critical for stability in dynamic scenes. Its formulation was originally introduced by Jiang et al. (2015) and is a key factor in more modern MPM implementation. More recently, the Moving Least Squares MPM variant (MLS-MPM), described in SIGGRAPH Talks by Zhu et al (2018), simplifies the MPM pipeline while improving the performance and flexibility. These ideas are further condensed in a compact implementation by Yuanming Hu, demonstrating a full MLS-MPM simulation in under 88 lines of annotated C++ code.

Liu's 2009 thesis introduces an illumination model tailored for snow surfaces, capturing effects like microshadowing and surface sparkle. Hiller et al. (2024) extend this by modeling volumetric light transmission, allowing snow to exhibit its characteristic blue tint due to internal scattering. These models inspired our use of bi-directional path tracing to simulate the translucent and glittering behavior of snow in both shallow and dense regions.

The Taichi MPM project, made by Hu, provides not only implementation references but also very impressive visualizations and performance techniques that shaped the design of the pipeline.

3 MOVING LEAST SQUARES MPM

The Moving Least Squares Material Point Method (MLS-MPM) extends standard MPM by enhancing the accuracy of velocity transfers between particles and grid. Each particle maintains an affine velocity field in addition to its velocity, allowing improved simulation of rotation and shear. This approach builds upon the Affine Particle-In-Cell method (APIC) introduced by Jiang et al. (2015), and is formalized in the MLS-MPM paper by Zhu et al. (2018).

The deformation gradient **F** is updated as:

$$\mathbf{F}_{t+\Delta t} = (\mathbf{I} + \Delta t \cdot \mathbf{C}) \cdot \mathbf{F}_t$$

where:

- **F** is the deformation gradient matrix, tracking local shape changes of a particle.
- **I** is the 3×3 identity matrix.
- Δt is the simulation timestep.
- **C** is the affine velocity matrix derived from grid data.

4 SIMULATION PIPELINE

Our implementation follows the four-stage MPM pipeline described in Lynch (2022). Each simulation step consists of clearing the grid, transferring data from particles to grid (P2G), updating grid velocities under forces, and interpolating the updated data back to the particles (G2P).

Algorithm 1 MPM Simulation Step

```
Step grid.clear() Reset grid velocities and mass p2g() Transfer
particle data to grid grid.update() Apply gravity and boundary
logic g2p() Update particle velocities and positions
```

Particle-to-Grid (P2G)

In this stage, particles transfer mass, velocity, and internal stress to a $3 \times 3 \times 3$ grid neighborhood. Quadratic B-spline weights w are used for interpolation.

The Cauchy stress $\boldsymbol{\sigma}$ is derived using a compressible Neo-Hookean model:

$$\boldsymbol{\sigma} = \frac{1}{J}\left[\mu(\mathbf{F} - \mathbf{F}^{-T}) + \lambda \log J \cdot \mathbf{F}^{-T}\right] \cdot \mathbf{F}^T$$

where:

- μ, λ are Lamé parameters controlling shear and volumetric stiffnes.
- **F** is the deformation gradient.
- $J = \det(\mathbf{F})$ is the Jacobian determinant, measuring volume change.
- $\mathbf{F}^{-T}$ is the inverse transpose of **F**.

We then compute the particle's momentum contribution to the grid:

$$\mathbf{p}_{\text{grid}} = m(\mathbf{v} + \mathbf{C} \cdot \Delta\mathbf{x}) - \Delta t \cdot \boldsymbol{\sigma} \cdot \Delta\mathbf{x}$$

with:

- m as the particle mass.
- **v** as the particle velocity.
- **C** as the affine velocity matrix.
- $\Delta\mathbf{x}$ is the offset vector from the particle to the grid node.

Grid Update

Grid cell velocities are normalized by total cell mass and updated using:

$$\mathbf{v}_{\text{grid}} \mathrel{+}= \Delta t \cdot \mathbf{g}$$

where **g** is the gravity vector. Grid velocities are also clamped based on a rotated boundary volume, allowing non-axis-aligned domains for scenes like sloped terrain and avalanches.

Grid-to-Particle (G2P)

After updating the grid, data is interpolated back to the particles using the same weights as in P2G. The affine matrix **C** is reconstructed from the weighted outer products of grid velocity and spatial offsets. This enables particles to capture local flow details like shear and rotation.

The updated velocity field is used to advance the deformation gradient:

$$\mathbf{F} \leftarrow (\mathbf{I} + \Delta t \cdot \mathbf{C}) \cdot \mathbf{F}$$

To simulate snow-like plasticity, we apply SVD-based clamping of **F**. The singular values σ_i of **F** are limited:

$$\sigma_i \leftarrow \text{clamp}(\sigma_i, \sigma_{\min}, \sigma_{\max})$$

before recomposing **F** as:

$$\mathbf{F} = \mathbf{U} \cdot \text{diag}(\sigma) \cdot \mathbf{V}^T$$

Time Integration

Particle positions are finally updated using forward Euler integration:

$$\mathbf{x}_{t+\Delta t} = \mathbf{x}_t + \Delta t \cdot \mathbf{v}$$

This framework supports effects such as compaction, rolling, and fracture while remaining stable under complex interactions.

5 SIMULATION PARAMETERS

Our simulation is governed by a set of tunable parameters that control both the global simulation behavior and the material properties of snow. These parameters influence the simulation's accuracy, stability, and physical realism.

5.1 Simulation Controls

Timestep dt: Controls how far forward the simulation advances per step. Smaller timesteps increase stability and precision but require more iterations. In our implementation, **dt** is chosen to maintain a balance between speed and stability, particularly important when simulating fast-moving snow or high-deformation events like avalanches.

Grid Cell Size dx: Defines the spatial resolution of the grid. A smaller **dx** increases the simulation's ability to capture fine details, such as snow compression patterns and surface variation, at the cost of higher computational overhead. In our setup, **dx** = 1.0 by default, with rendering scaled accordingly.

Grid Size: Specifies the extent of the simulation domain in all three spatial dimensions. A larger grid enables the simulation of expansive environments (e.g., slopes or large snow fields), while a smaller grid is more efficient for localized interactions. Grid size also indirectly constrains particle motion and interaction.

5.2 Material Properties

Shear Modulus μ: Represents the material's resistance to shear deformation (i.e., shape changes at constant volume). A higher μ results in stiffer snow that resists stretching and tearing. For example, we set a relatively low μ for soft, fresh snow and a higher μ for compacted snow.

Bulk Modulus λ: Governs the resistance to volumetric compression. This parameter ensures that snow behaves realistically under pressure—denser packing in impact zones and gradual compaction under weight. Higher values of λ yield more incompressible snow, mimicking wetter or denser snowpacks.

Plasticity Parameters

Plasticity is optionally enabled in our simulation to capture snow's ability to permanently deform under stress, a key characteristic for compaction and accumulation. This is implemented by clamping the singular values of the deformation gradient **F** during the Grid-to-Particle phase.

Stretch Limits ($\sigma_{\min}$, $\sigma_{\max}$): Define thresholds for the deformation gradient's singular values. Values outside this range are clipped, allowing the material to yield plastically. This helps simulate effects such as snow clumping and irreversible compression without instability.

EnablePlasticity: A boolean switch to turn plasticity modeling on or off. When disabled, the material behaves purely elastically. Some types of snow are not as rigid.

5.3 Gravity

Gravity g is applied during the Grid Update stage and is essential for all snow motion, particularly falling particles, rolling snow, and avalanches. It is defined as a constant acceleration vector (typically $\mathbf{g} = (0, -9.8, 0)$), and affects every grid cell's velocity proportional to mass.

5.4 Tuning Parameters

Each parameter plays a critical role in producing believable snow behavior. For example, inappropriate combinations of μ, λ, and **dt** can lead to artifacts such as excessive bouncing, non-physical stretching, or numerical explosions. Tuning these parameters is essential not only for stability but for achieving material realism in different snow types—from soft powder to icy crust.

6 DEFORMATION & COMPRESSION

A key visual and physical effect of snow is its ability to deform under weight and retain compressed shapes. In our simulation, deformation is captured by the evolution of the particle deformation gradient **F**, which is updated each frame based on the local grid velocity field:

$$\mathbf{F} \leftarrow (\mathbf{I} + \Delta t \cdot \mathbf{C}) \cdot \mathbf{F}$$

Here, **C** is the affine velocity field from the APIC formulation, and **I** is the identity matrix. This formulation allows for realistic modeling of both linear and rotational deformations.

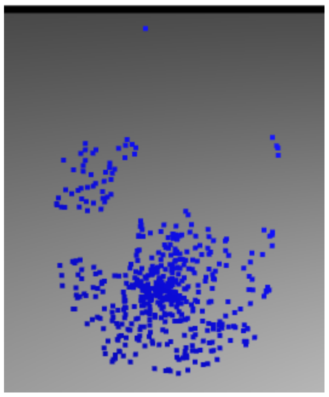

Figure 2: 2D Broken snowball after deformation

To simulate snow's compressibility, we apply an optional plasticity model. This involves computing the Singular Value Decomposition (SVD) of **F** and clamping the resulting stretch values to a specified range. By limiting how much a particle can stretch or compress, the material accumulates permanent deformation, leading to visual effects like packing and densification.

Compression occurs naturally during particle-grid interactions and is reinforced by stress computations derived from the Neo-Hookean model, governed by shear and bulk modulus values (μ, λ). This enables the simulation to reflect soft, fluffy snow that compacts into denser forms under pressure.

7 SNOW INTERACTION

Our simulation supports multiple forms of snow interaction, where individual particles or regions of snow dynamically influence each other. These include:

Snow-on-Snow Collisions: When falling snow particles land on accumulated regions, momentum and stress are transferred through the grid, allowing the incoming particles to embed, compress, or roll over the surface naturally.

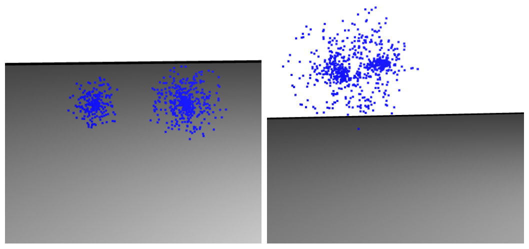

Figure 3: Two snowballs colliding with each other

Packing: As more snow accumulates in a region, increased stress leads to compaction. This behavior is enhanced when plasticity is enabled, resulting in permanent deformation and visual consolidation of snow.

Rolling and Redistribution: By applying frictional boundary conditions on the grid and allowing rotated slopes, snow is able to slide or roll under the influence of gravity. This makes interactions feel more grounded and allows natural terrain adaptation.

These interaction effects are entirely emergent from the core simulation rules, made possible by grid-mediated particle communication and consistent transfer of stress, velocity, and deformation.

7.1 Avalanche

To simulate avalanche-like behavior, we take advantage of grid-level transformations. The grid itself remains axis-aligned internally, but particle positions are evaluated against a rotated and translated boundary. This allows us to place the effective simulation domain on a slope, enabling gravity to naturally pull snow downward along the inclined surface.

Rotated Boundaries: The boundary of the grid is defined in world space using an inverse rotation matrix and translation vector:

$$\text{local\_pos} = \mathbf{R}^{-1}(\mathbf{x}_{\text{world}} - \mathbf{T})$$

Particles and cells are clamped or invalidated if they fall outside this rotated bounding box. By tuning the rotation angle and offset, we simulate inclined terrain without modifying the underlying grid structure.

Surface Friction: The simulation applies surface friction and slope-aware behavior by rotating the simulation grid and enforcing velocity adjustments at the boundaries. When a cell lies outside the designated boundary, its velocity is projected to remove the normal component, simulating contact with a surface. Friction is then applied to the remaining tangential velocity—low speeds are stopped entirely to simulate static friction, while higher speeds are gradually reduced using a dynamic friction model. This allows snow to build up momentum as it moves downslope, while still compacting and accumulating realistically, supporting behaviors like avalanches and rolling.

Together, these methods enable a dynamic avalanche effect, where snow builds up, collapses, and flows with emergent behavior, driven purely by simulation parameters and boundary geometry.

8 BI-DIRECTIONAL PATH TRACING

The rendering of a scene is done through bi-directional path tracing, as described in *Rendering Participating Media with Bidirectional Path Tracing*. This method involves tracing rays from the camera, through each pixel to render the scene, but additionally involves tracing rays in the opposite direction, from light sources into the scene.

8.1 Tracing From Sources

The number of bounces from the light source is specified by the user. Then, for each initial path trace, the light source will trace the desired number of bounces into the scene, creating a representative light source at each intersection, called a light point. These light points have their intensity scaled by the material properties of the hit but are otherwise treated as light sources for that path trace.

8.2 Additional Rays

Due to the utilization of path tracing (rather than other variations of ray tracing), only one random sample is chosen for each type of ray for each initial path trace. That is, upon hitting a surface, only one random shadow ray is cast per light source, and if reflective, only one reflection ray is cast. Then, instead of using multiple random samples per hit, a variety of initial casts are performed per pixel. This gives similar quality renderings to other ray-tracing variations while significantly reducing the total number of rays cast. This is due to the contribution of a ray decreasing at each bounce, meaning front loading the rays gives a higher impact for the same, or better, performance.

9 SURFACE RENDERING

The first integration between the physics simulation and rendering implementation involves how to render the MPM system as an object. The MPM representation contains a collection of particles that act as representatives of the general system. This means that one particle corresponds to a number of actual snowflakes. In addition, the overall grid representation exists and can provide additional information about the simulation. Our intersection detection and normal calculations have gone through a variety of implementations, with further improvement planned for the future. Below is the current status.

9.1 MPM Intersection Detection

In order to render the MPM system using ray casting, an intersection function needs to be defined for it. This acts as a method of determining if a ray has hit the system and, if so, where. Due to the unique representation of the system (unlike a simple mesh or shape equation), a clever intersection detection is needed. Currently, each particle is treated as the center of a mass of snow. To check for intersection, the closest point along the ray to each particle is found. If this point is within a certain threshold, we note that the ray intersects with that mass of snow. We then find the intersected particle closest to the camera and return it's hit as the best. This solution is naive and leads to the render appearing as a collection of spheres, and this is one of the biggest opportunities for future work.

9.2 Normal Calculation

Along with the hit position, the ray needs to know the hit normal to properly shade the pixel. Calculation of normal is greatly dependent on intersection implementation. Due to the particle based intersection present at the moment, normals are calculated to be the normalized direction from the actual particle that was hit to the point in space the hit was registered at. This further attributes to the collection of spheres render, giving accurate normals for that implementation. As intersection implementation is improved, normal calculation will need to be modified to match.

9.3 "Sparkle" Effect

The sparkle of snow is one of the key effects we desired to represent in our system. The physical explanation attributes sparkle to the small-scale structure of snowflakes. The snow surface is made up of ice crystals with flat surfaces that point in all directions. Therefore, a number of these crystals are angled just right to reflect and refract light toward a viewer, causing these crystals to appear significantly brighter than the surrounding snow in the proper conditions. To represent this, variation needs to be applied to the normals of the snow. Currently, this is done naively by adding the noise at render time. This creates inconsistent and inaccurate sparkle for still scenes. However, given the current state of the rest of the system, this works and gives accurate results. Upon tracing for a pixel, if an MPM intersection is found and the normal points directly to a (main) light source, a sparkle color is returned instead.

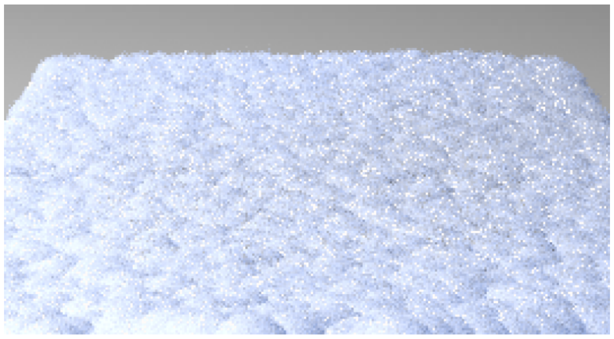

Figure 4: White sparkle on a relatively flat sheet of snow

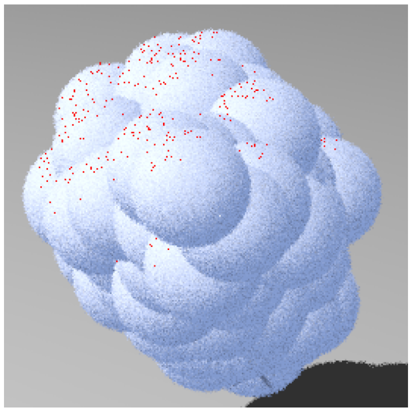

Figure 5: Red visualizations of sparkle points on a snowball

10 SUBSURFACE RENDERING

Once an intersection with an MPM particle has been determined, the rendering doesn't stop there. A significant portion of light scatters into the snow, illuminating it from within. This is explicitly calculated while path-tracing.

10.1 Subsurface Scattering

After a ray has determined it hit an MPM particle, it creates a subscatter ray to check if any contribution may come from light within the snow. The depth of this test is preconfigured. Until escaping or reaching the depth, a ray is cast from the hit point toward the actual MPM particle hit. This simulates the light bouncing into the mass of snow (or, rather, finding the contribution from light that may be coming from the mass to the intersection point). This cast has a slight random offset so the particle isn't directly hit every time, and the intersection has a much smaller threshold distance than the initial hit detection, giving the ray a chance to escape. If anything other than an MPM particle is hit with the depth, the ray is noted as "escaped".

10.2 Tinting

Once it is known a ray has escaped, it's contribution can be added to the color of the pixel. The physical properties of real snow allow shorter frequencies of light to escape at a higher level than lower frequencies, leading to the blue-ish tint of snow. Rather than simulating multiple frequencies, our solution takes this fact into account, and tints the contribution of the escaped ray to match this blue-ish tint.

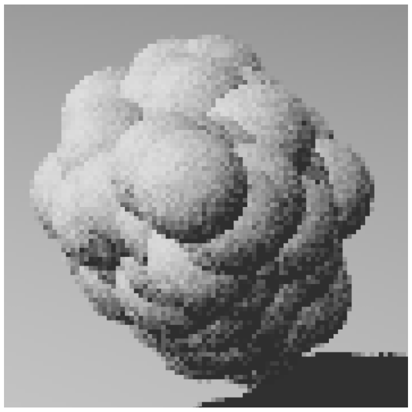

Figure 6: Visualization of snow *without* contribution from subsurface scatter tinting

11 COMBINED PHYSICS RENDERING

Our system integrates a physically-based simulation of snow with a bidirectional path tracing renderer. While the simulation runs in a particle–grid loop using MLS-MPM, rendering is performed offline to ensure high visual fidelity. This tight coupling allows visualizations to reflect not just particle motion, but also surface phenomena and optical snow properties.

11.1 Simulation–Render Synchronization

Each simulation step advances the snow state using MLS-MPM. The renderer operates synchronously with this loop: after each simulation frame is computed, particle positions are rendered and saved to disk. This ensures that all visual outputs are physically consistent with the ongoing simulation.

The rendering system reads particle positions, transforms them from world space to normalized device coordinates, and displays them as shaded points. However, for high-quality output, we use a full-frame raytracing pass to capture realistic lighting, including shadows, reflections, and subsurface color attenuation.

11.2 Frame Export Pipeline

To generate high-resolution frames, we extract the pixel data directly from the OpenGL framebuffer. This involves reading RGB values using `glReadPixels`, vertically flipping the data to match PPM image format conventions, and writing it to disk. Each image is stored sequentially to build an animation.

The export process:

(1) Capture the framebuffer after rendering.
(2) Flip image data vertically to align with raster scan order.
(3) Write to disk in binary PPM format, indexed by frame number.

11.3 Raytraced Animation Mode

The renderer supports a special animation mode (`render_mpm_movie`) that disables interactive simulation and instead steps the system one frame at a time. After each `step()`, raytracing is performed using a subdivided sampling grid, and the resulting frame is saved. This ensures consistent rendering at each simulation state, with complete lighting including indirect effects. While slow, this method produces photorealistic results suitable for video export.

11.4 Coupling Simulation and Optics

By tying the render pipeline directly into the simulation loop, we are able to visualize physically rich effects like compaction, avalanche behavior, and snow–surface interaction. This integration makes it possible to demonstrate both mechanical and visual aspects of snow in the same scene, with consistent motion and lighting between frames.

12 ADDITIONAL FEATURES

Beyond the core simulation and rendering pipeline, our implementation includes several enhancements to improve flexibility, usability, and performance.

12.1 Custom scene creation

To support a wide variety of physical phenomena, the system supports multiple configurable MPM objects per scene. These can represent different snow volumes with distinct shapes, positions, and initial properties. Additionally, scenes can be procedurally constructed using geometric primitives such as snowballs, sheets, and inclined planes to create scenarios like collisions or avalanches.

12.2 Interactive Controls

For testing and demonstration purposes, interactive controls are available during runtime:

- `a` – Start the physics simulation
- `space` – Advance simulation by a single step
- `c` – Reset simulation state
- `m` – Enable movie rendering mode
- `r` – Trigger a full bidirectional path traced render

These controls allow for both debugging and demonstration without recompiling or editing source files.

12.3 Performance Optimization

To accelerate computation, the most intensive parts of the simulation pipeline, including Grid Clear, Grid Update, and Grid-to-Particle (G2P) transfer are fully parallelized using multithreading. Work is split across available CPU cores using a thread pool, providing scalable performance for high-resolution simulations.

13 LIMITATIONS & CHALLENGES

Achieving both physical accuracy and interactive performance in a snow simulation is a significant challenge. Snow is inherently complex, exhibiting behaviors such as packing, flowing, and fracturing, which require careful modeling of its mechanical properties. Implementing these behaviors necessitates high shear and bulk modulus values, which in turn demand smaller time steps to maintain stability. This results in longer simulation times and reduces the feasibility of real-time interaction.

Plasticity, although essential for realistic snow compaction and flow, adds additional overhead due to singular value decomposition (SVD) and clamping operations applied to each particle's deformation gradient. While these operations are necessary to simulate snow's characteristic compression behavior, they also contribute to computational expense.

Another challenge lies in parameter tuning. Small variations in the simulation timestep, stiffness constants, or grid resolution can lead to significantly different outcomes, making it difficult to find a stable yet visually plausible configuration. Additionally, the implementation relies on a full 3D particle grid system and parallelized updates, which, while optimized, still struggle to meet real-time rendering and simulation demands under complex conditions.

Additionally, the representation of snow as a collection of grid cells and particles leads to difficulty in the calculation of intersections and normals while path tracing. This lead to a variety of implementations being tested while working towards the current implementation. Even then, the current implementation is far from perfect and still overgeneralizes the behavior rather than being physically accurate.

14 FUTURE WORK

There are several avenues for improvement. First, simulation speed could be addressed with more aggressive optimizations, such as GPU acceleration or adaptive grid resolution, where computational effort is focused on regions with higher activity. Second, a more systematic parameter tuning framework could help users define material behavior more intuitively while preserving physical realism.

On the rendering side, real-time approximations for effects such as subsurface scattering and sparkle could allow us to achieve visually compelling results at interactive frame rates. In the opposite direction, improved methods to calculate intersections, normals, and improved subsurface scattering contributions would lead to results with greater accuracy.

Support for snow melting, freezing, or transitioning between material states could extend the system's realism and range of applications.

Physically, more advanced plasticity models and fracture-aware constitutive models could enhance the realism of snow fracture, shear bands, or granular avalanche behavior. Additional improvements to the interaction between snow and rigid bodies including collision, friction, and accumulation would also elevate realism, especially in scenes with moving geometry.

15 BREAKDOWN OF WORK

This project, including research, brainstorming, implementation, testing, presentation, and reports took a collective 100 hours over 4-5 weeks between the two of us. Tre'Vaughn worked on the MLS-MPM implementation, from representation, to configuration, to simulation, as well as the combined physics rendering feature. Jaden worked on extending the raytracing implementation to utilize bi-directional path tracing and interface with the MPM simulation, as well as additional extensions for snow-specific optical features, such as sparkle and subsurface scattering.

16 CONCLUSIONS

This project presents a particle-based simulation framework for snow using the Moving Least Squares Material Point Method (MLS-MPM), with tight integration between physics and rendering. Through this system, we are able to simulate and visualize complex snow behavior including deformation, compression, and interaction effects such as rolling or avalanche formation. Although the system currently operates at offline speeds, it provides a solid foundation for exploring realistic snow dynamics and rendering in a unified architecture.

This system was designed with extensibility in mind. It sets up a framework for both physically simulating and rendering snow, while providing support for many physical and optical effects that make snow unique. On the simulation side, additional tuning can be done to better represent certain features and add support for others. On the rendering side, each section can be easily improved upon to increase realism, and additional effects can be slotted in as they come up.

The combination of a physically grounded simulation pipeline with flexible rendering tools demonstrates how snow can be treated as a dynamic and optically rich material. With further refinement, this approach has the potential to support interactive environments, cinematic visual effects, and experimental scientific visualizations of granular snow-like materials.

ACKNOWLEDGMENTS

To Professor Barb Cutler, for providing the code for connecting the ray tracing algorithm to OpenGL for rendering.

BLOOPER IMAGES

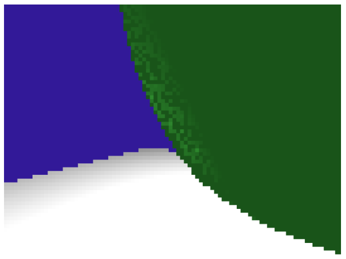

Figure 7: First attempt at rendering MPM particles, using a green material to visualize. Neglected to scale from MPM coordinate space to world space, leading to a giant mass centered at (32, 32, 32).

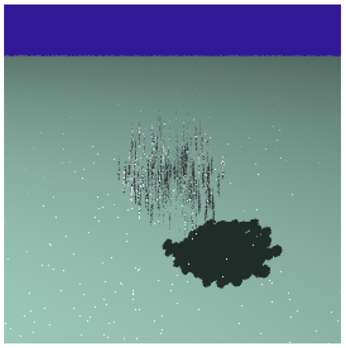

Figure 8: Only checking for intersection proximity in one axis, leading to "strips"

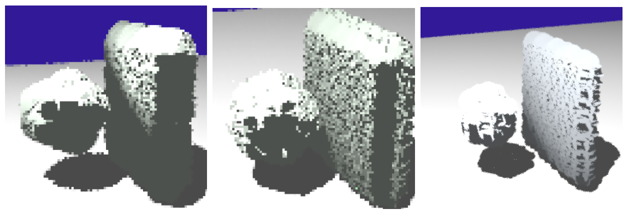

Figure 9: Progression of normal calculation implementations prior to the current solution.

REFERENCES

LYNCH, N.
A Practical Guide to the Material Point Method (MPM). 2022.
nialltl.neocities.org.
https://nialltl.neocities.org/articles/mpm_guide

STOMAKHIN, A., HOWES, R., SCHROEDER, C., CHUANG, E., TERAN, J., AND SELLE, A.
A Material Point Method for Snow Simulation. 2013.
ACM Trans. Graph. 32, 4, Article 102.
https://doi.org/10.1145/2461912.2461948

LIU, F.
An Illumination Model for Realistic Rendering of Snow Surfaces. 2009.
Master's Thesis, Linköping University, Sweden.
https://www.diva-portal.org/smash/get/diva2:292565/FULLTEXT01.pdf

LAFORTUNE, E.P., WILLEMS, Y.D.
Rendering Participating Media with Bidirectional Path Tracing. 1996.
Pueyo, X., Schröder, P. (eds) Rendering Techniques '96. EGSR 1996. Eurographics. Springer, Vienna.
https://doi.org/10.1007/978-3-7091-7484-5_10

HILLER, S., DROSKE, M., REIS, G., MEINL, M., KLEIN, R., AND DACHSBACHER, C.
Rendering the Bluish Appearance of Snow: When Light Transmission Matters. 2024.
IEEE Comput. Graph. Appl. 44, 1, 48–56.
https://doi.org/10.1109/MCG.2023.3307517

JIANG, C., SCHROEDER, C., SELLE, A., TERAN, J., AND STOMAKHIN, A.
The Material Point Method for Simulating Continuum Materials. 2016.
SIGGRAPH Course Notes.
http://mpm.graphics

ZHU, Y., HU, W., JIANG, C., AND BAILEY, M.
The Moving Least Squares Material Point Method (MLS-MPM). 2018.
ACM SIGGRAPH Talks.
https://www.yzhu.io/projects/siggraph18_mlsmpm_cpic/index.html

JIANG, C., SCHROEDER, C., SELLE, A., TERAN, J., AND STOMAKHIN, A.
The Affine Particle-In-Cell Method. 2015.
ACM Trans. Graph. 34, 4, Article 51.
https://www.math.ucla.edu/~jteran/papers/JSSTS15.pdf

HU, Y.
MLS-MPM in 88 Lines of C++.
https://github.com/yuanming-hu/taichi_mpm/blob/master/mls-mpm88-explained.cpp

HU, Y.
Taichi MPM Implementation, Slides and Notes.
https://github.com/yuanming-hu/taichi_mpm/