MPM Simulation With Rigid Body Interactions

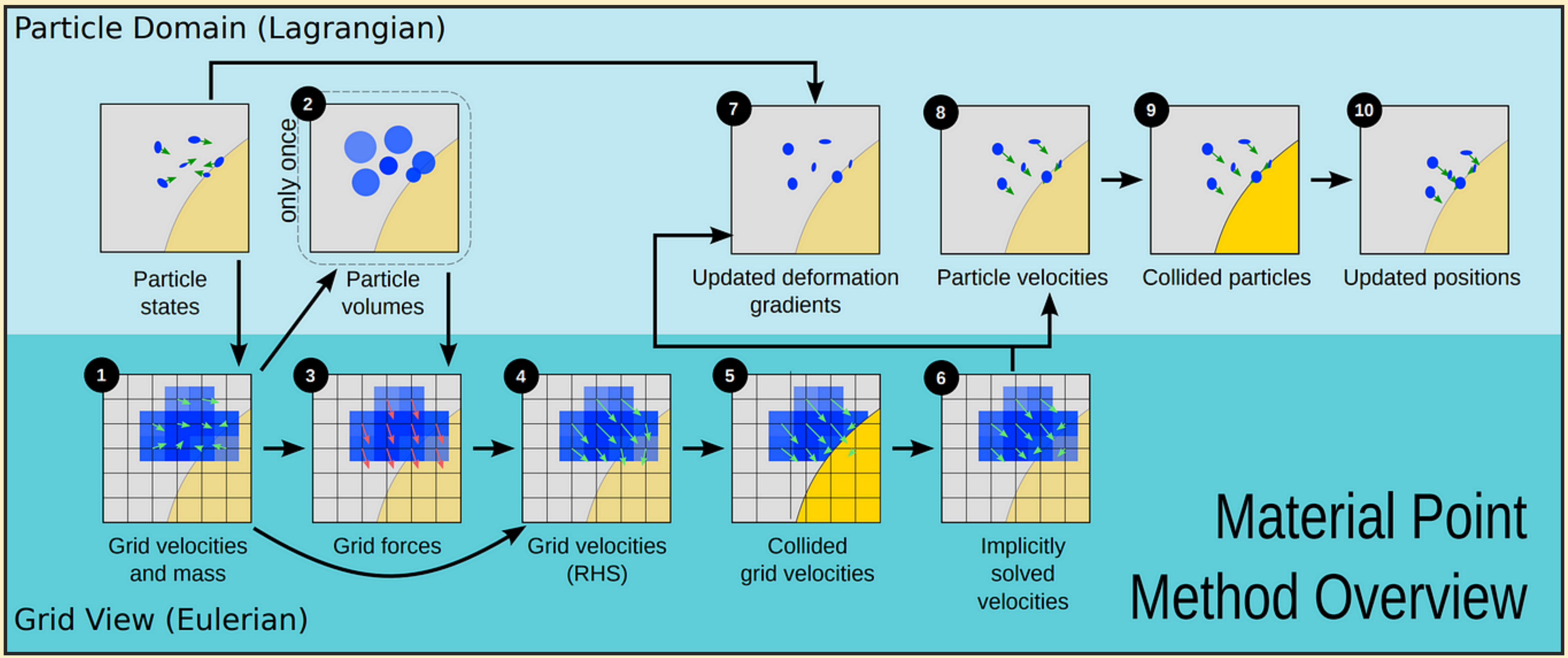
Žiga Kovačič, Evan Zhang

CS 5643, Final Project

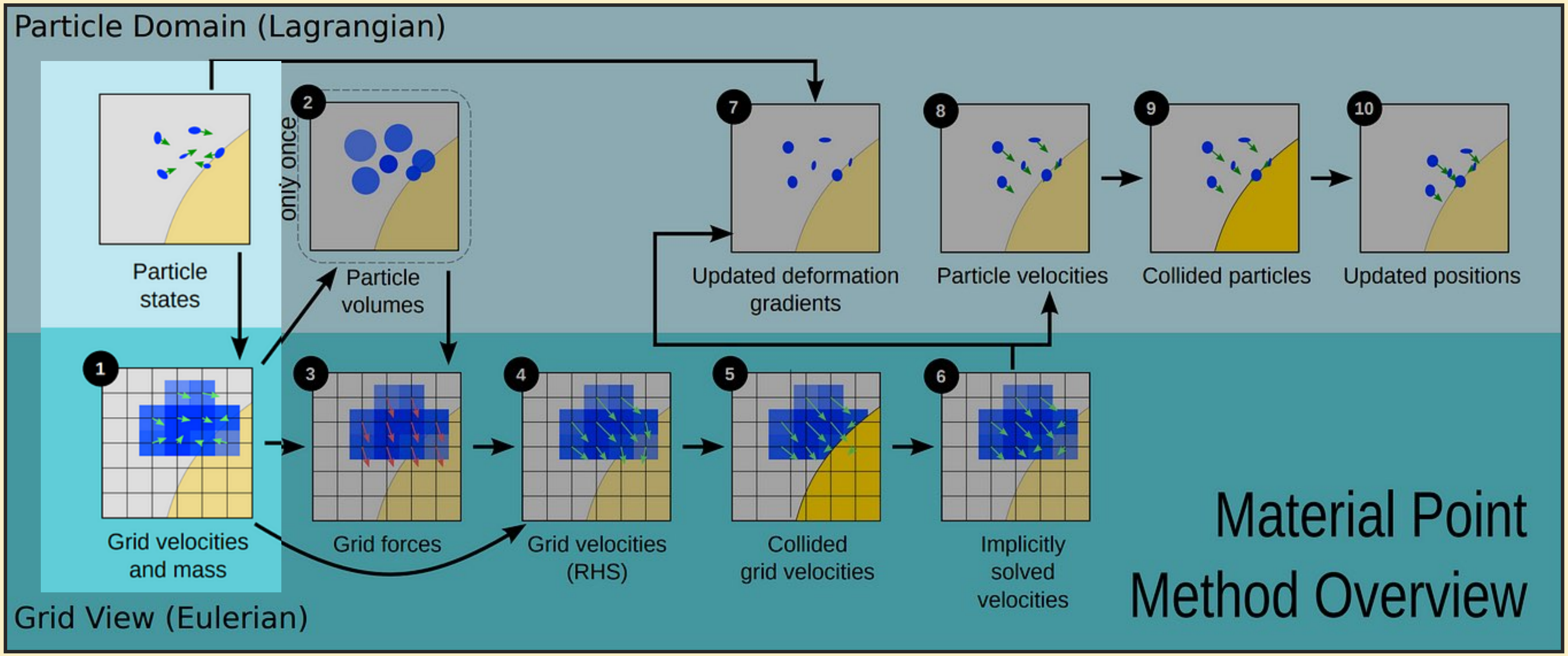
Background

Recall: MPM Algorithm

Hybrid Lagrange-Eulerian simulation framework



MPM to MLS-MPM



$$m_i^n = \sum_p m_p w_{ip}^n$$

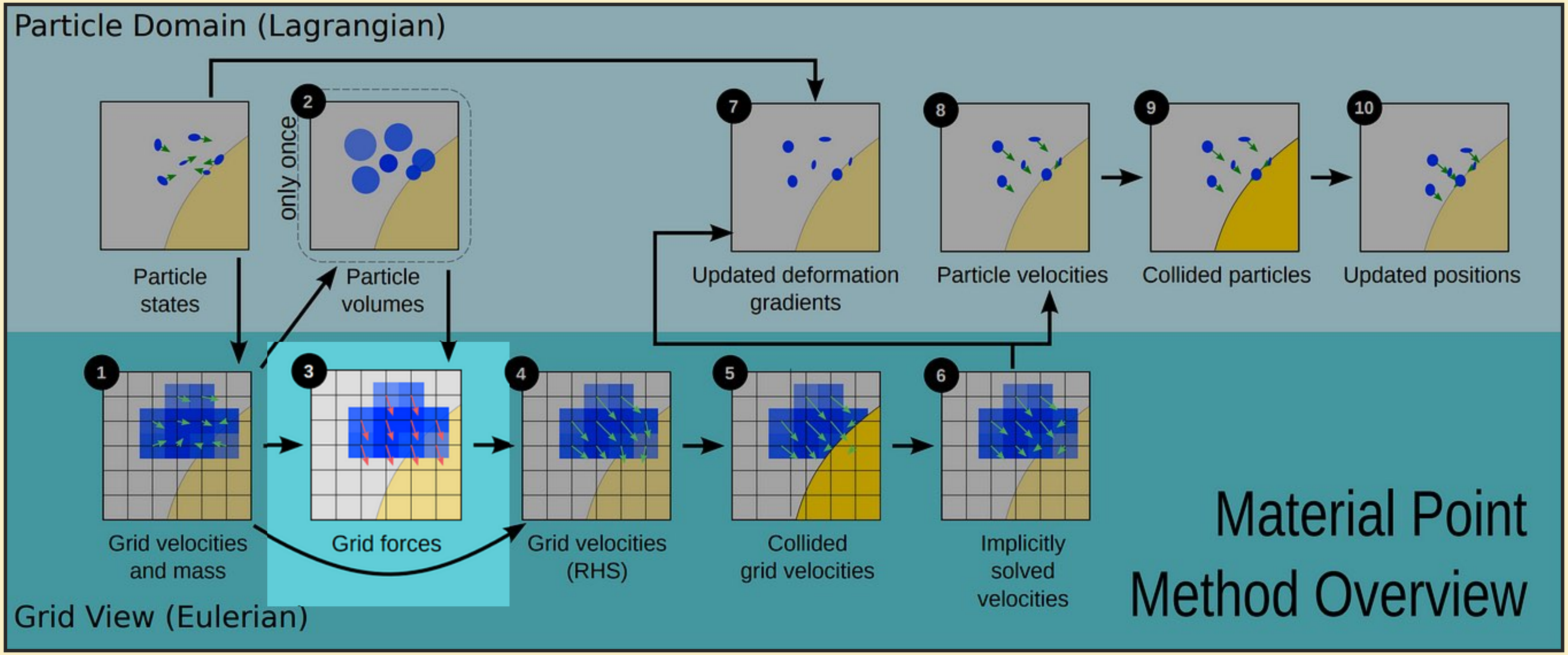
$$\mathbf{v}_i^n = \frac{1}{m_i^n} \sum_p \mathbf{v}_p^n m_p w_{ip}^n$$

$$m_i = \sum_p m_p w_{ip}$$

APIC transfer

$$\mathbf{v}_i = \sum_p m_p \left(\mathbf{v}_p + \mathbf{C}_p (\mathbf{x}_i - \mathbf{x}_p) \right) w_{ip}$$

MPM to MLS-MPM

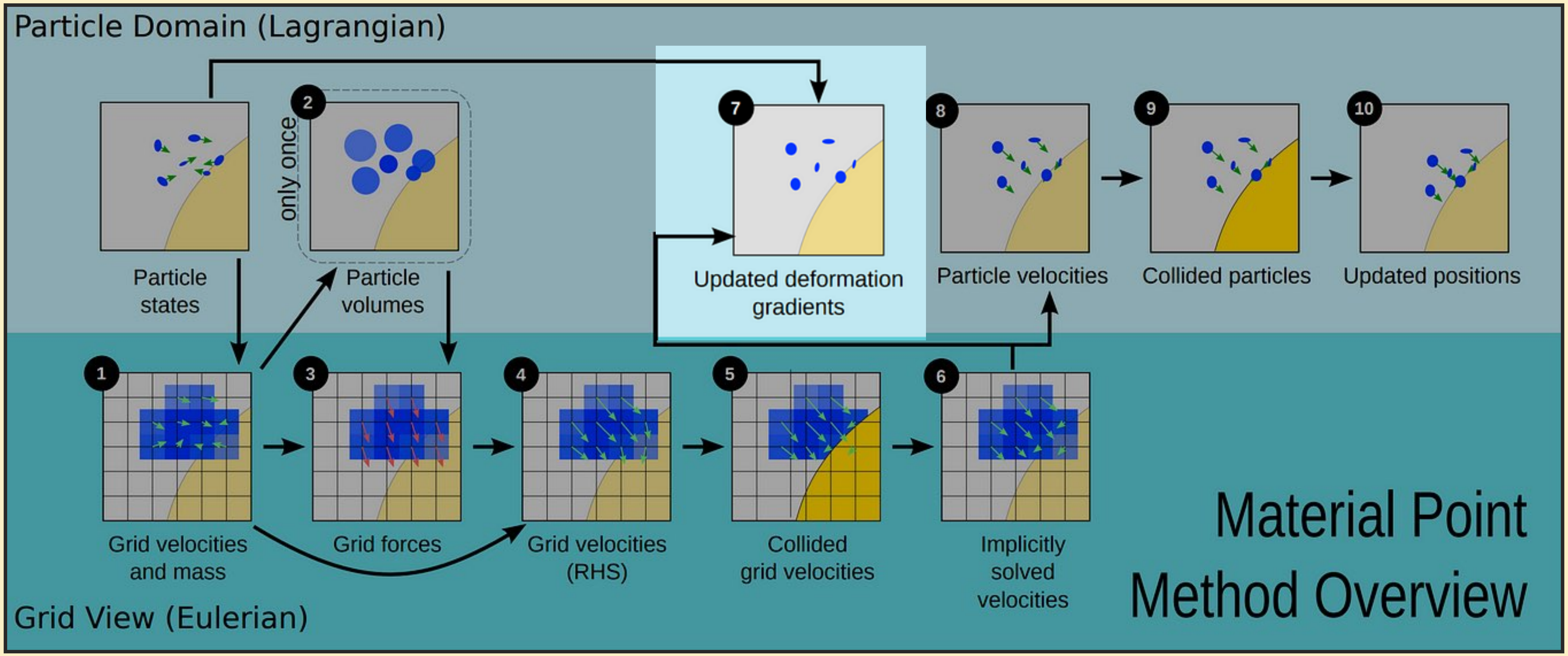


$$\mathbf{f}_i(\hat{\mathbf{x}}) = -\sum_p V_p^n \boldsymbol{\sigma}_p \nabla N_i(x_p^n) \quad \longrightarrow \quad \mathbf{f}_i = -\sum_p N_i(\mathbf{x}_p^n)\, V_p^n\, \mathbf{M}_p^{-1} \boldsymbol{\sigma}_p\, (\mathbf{x}_i^n - \mathbf{x}_p^n)$$

$\nabla N_i(x_p^n)$ is slow!

No gradient!

MPM to MLS-MPM



$$F_p^{n+1} = (I + \Delta t \nabla v_p^{n+1}) F_p^n \quad \rightarrow \quad F_p^{n+1} = (I + \Delta t C_p^{n+1}) F_p^n$$

$\nabla_x v_p^{n+1} = \sum_i \hat{v}_i^{n+1} \nabla N_i(x_p^n)^\top$ is slow!

Reuse C_p

MPM to MLS-MPM

Unification of the deformation gradient and affine velocity field

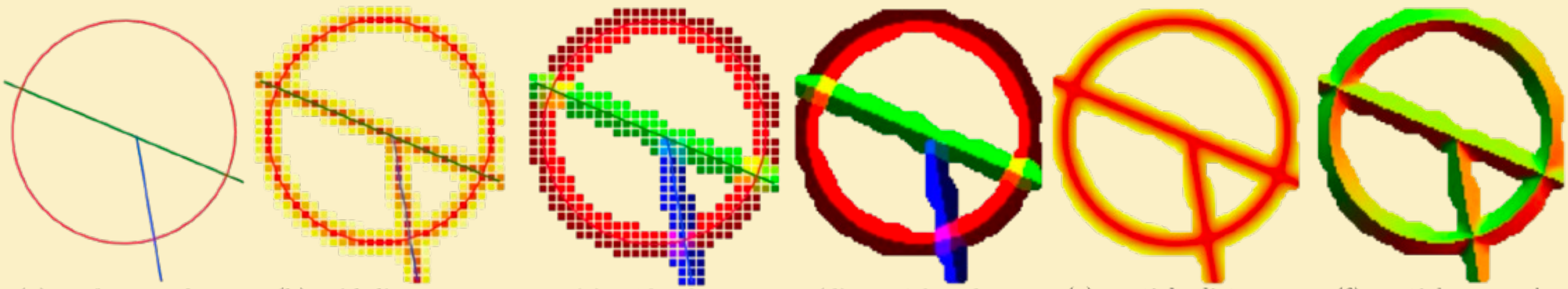
Shape/Test function	B-spline	MLS Shape function weighted by B-spline
Lumped mass matrix	$m_i^n = \sum_p m_p \omega_{ip}$	
APIC P2G Momentum Contribution	$m_p \mathbf{C}_p^n (\mathbf{x_i} - \mathbf{x_p}) \omega_{ip}$	
Stress Momentum Contribution	$\Delta t V_p^0 \frac{\partial \Psi}{\partial \mathbf{F}} (\mathbf{F}_p^n) \mathbf{F}_p^{nT} \nabla \omega_{ip}$	$\frac{4}{\Delta x^2} \Delta t V_p^0 \frac{\partial \Psi}{\partial \mathbf{F}} (\mathbf{F}_p^n) \mathbf{F}_p^{nT} (\mathbf{x}_i - \mathbf{x}_p) \omega_{ip}$
APIC G2P Affine Velocity Reconstruction	$\mathbf{C}_p^{n+1} = \frac{4}{\Delta x^2} \sum_i v_i (\mathbf{x}_i - \mathbf{x}_p) \omega_{ip}$	
Velocity Gradient Evaluation	$\nabla \mathbf{v}_p^{n+1} = \sum_i \boldsymbol{v}_i^{n+1} (\nabla w_{ip}^n)^T$	$\nabla \mathbf{v}_p^{n+1} = \mathbf{C}_p^{n+1}$
Deformation Gradient Update	$\mathbf{F}_p^{n+1} = \left(\mathbf{F} + \Delta t \nabla \mathbf{v}_p^{n+1}\right) \mathbf{F}_p^n$	

CPIC Speedrun

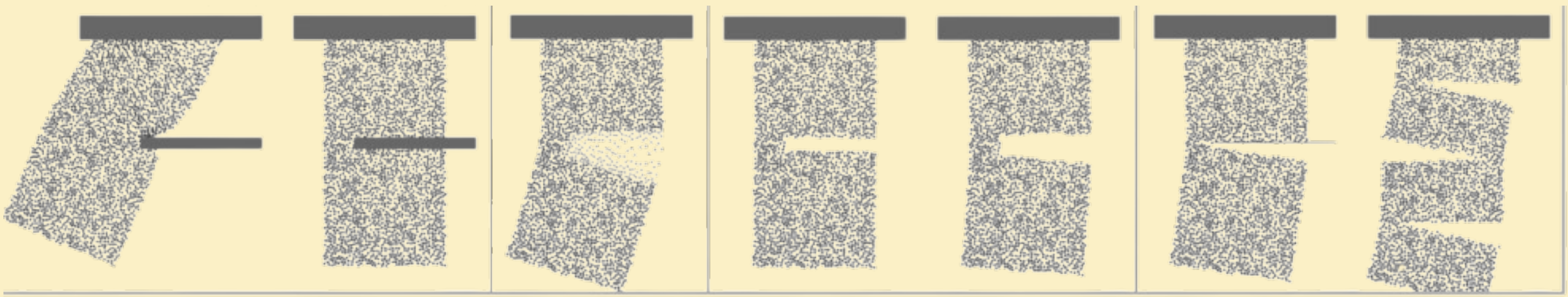
Sharp discontinuities

Introduce **“Compatible”** particle-cell pairs

Data transfer **restricted** to “Compatible” pairs

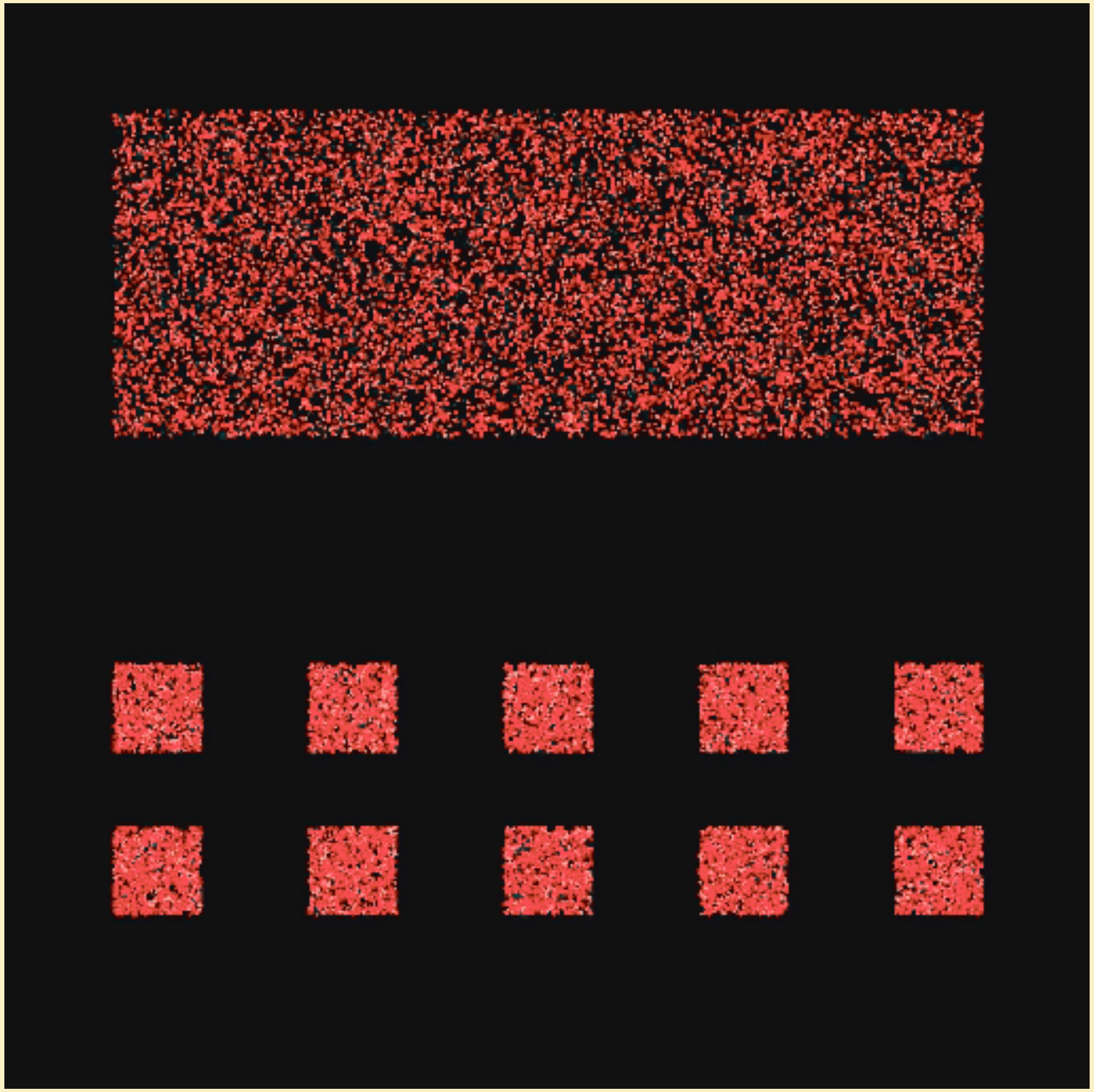


(a) surface mesh (b) grid distance (c) grid color (d) particle color (e) particle distance (f) particle normal



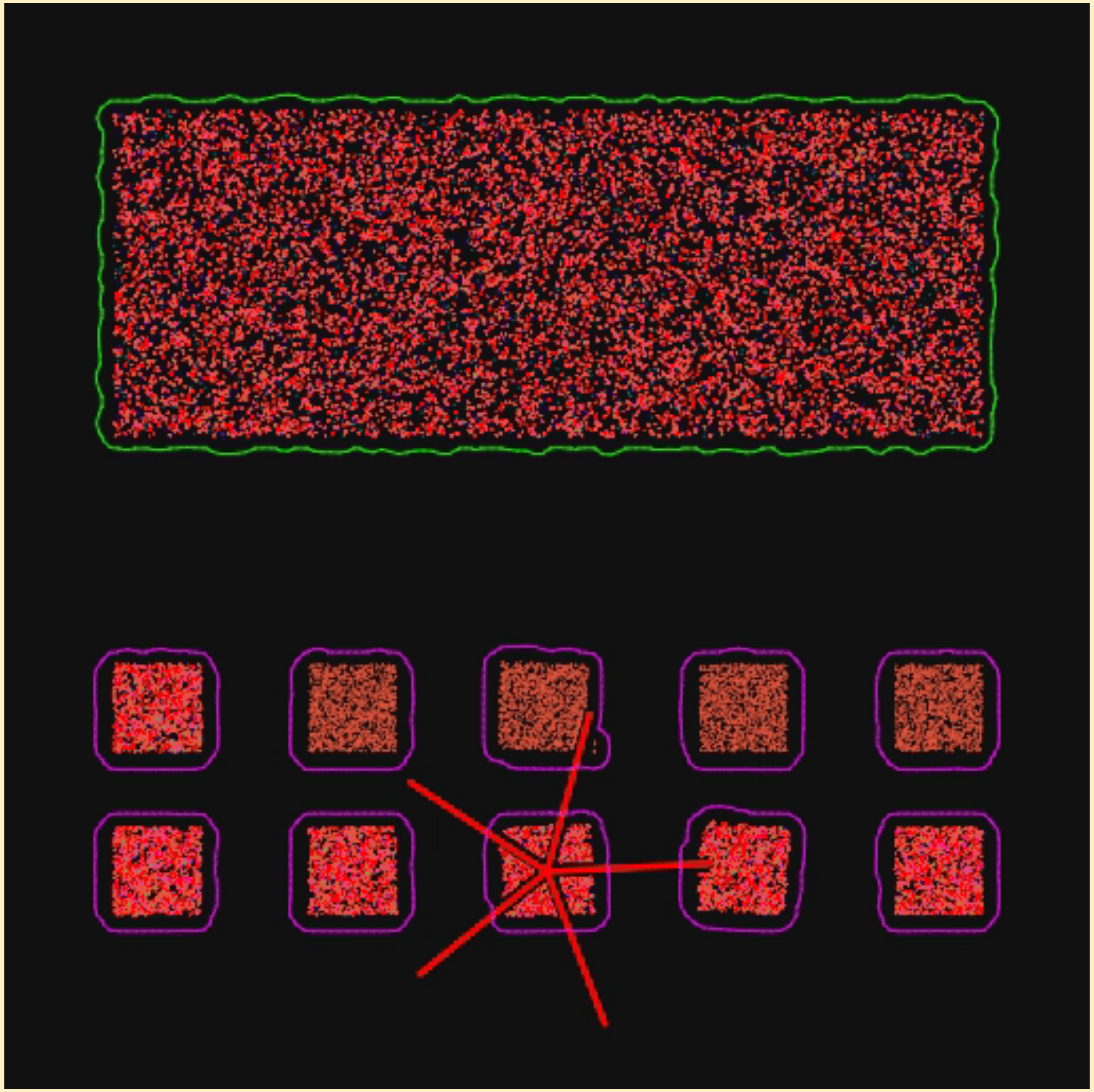
Progress: Part 1

MLS-MPM | 2D



Jello & Fluid interactions, wall collisions.

Surface Extraction: Marching Squares

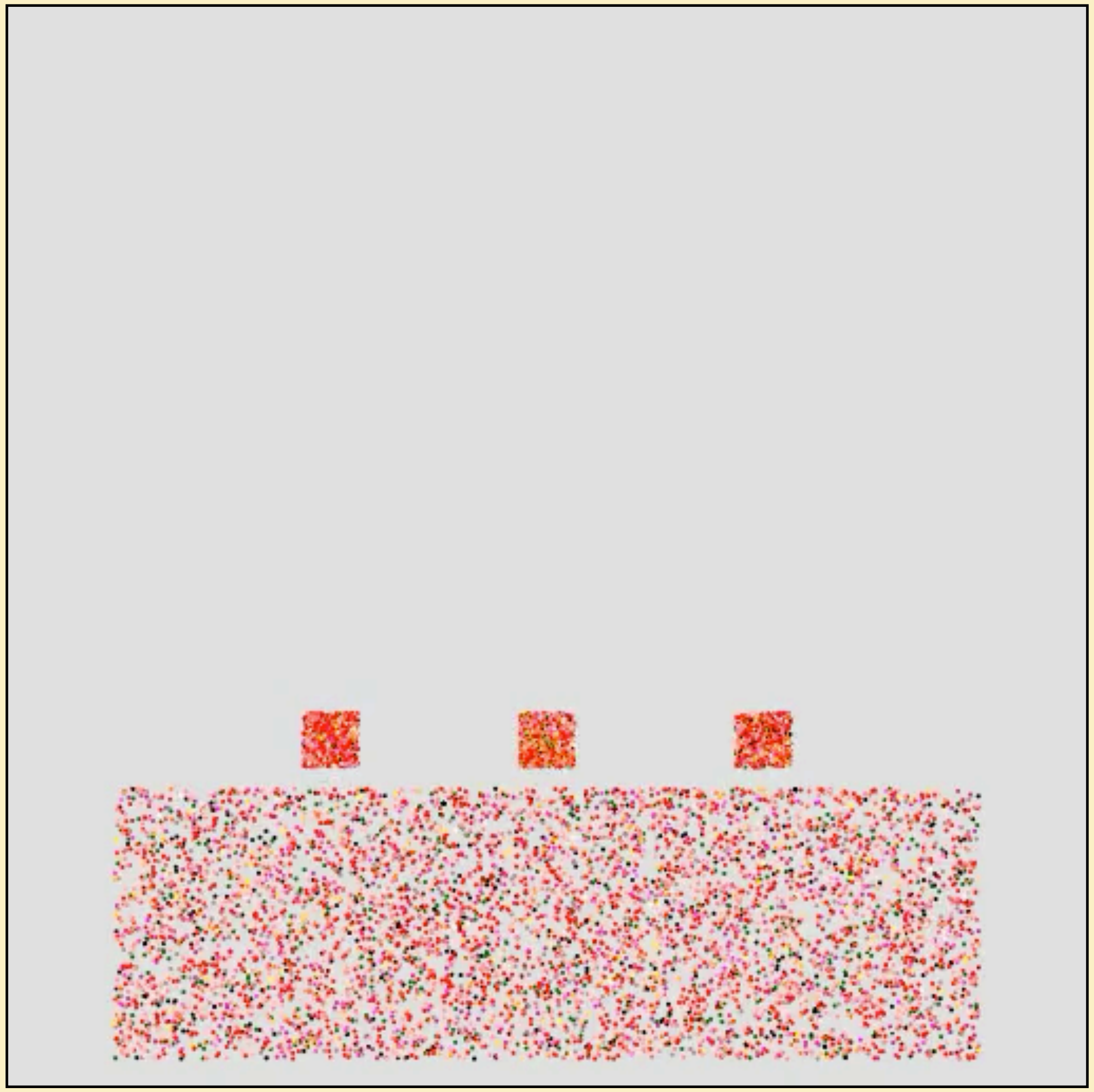


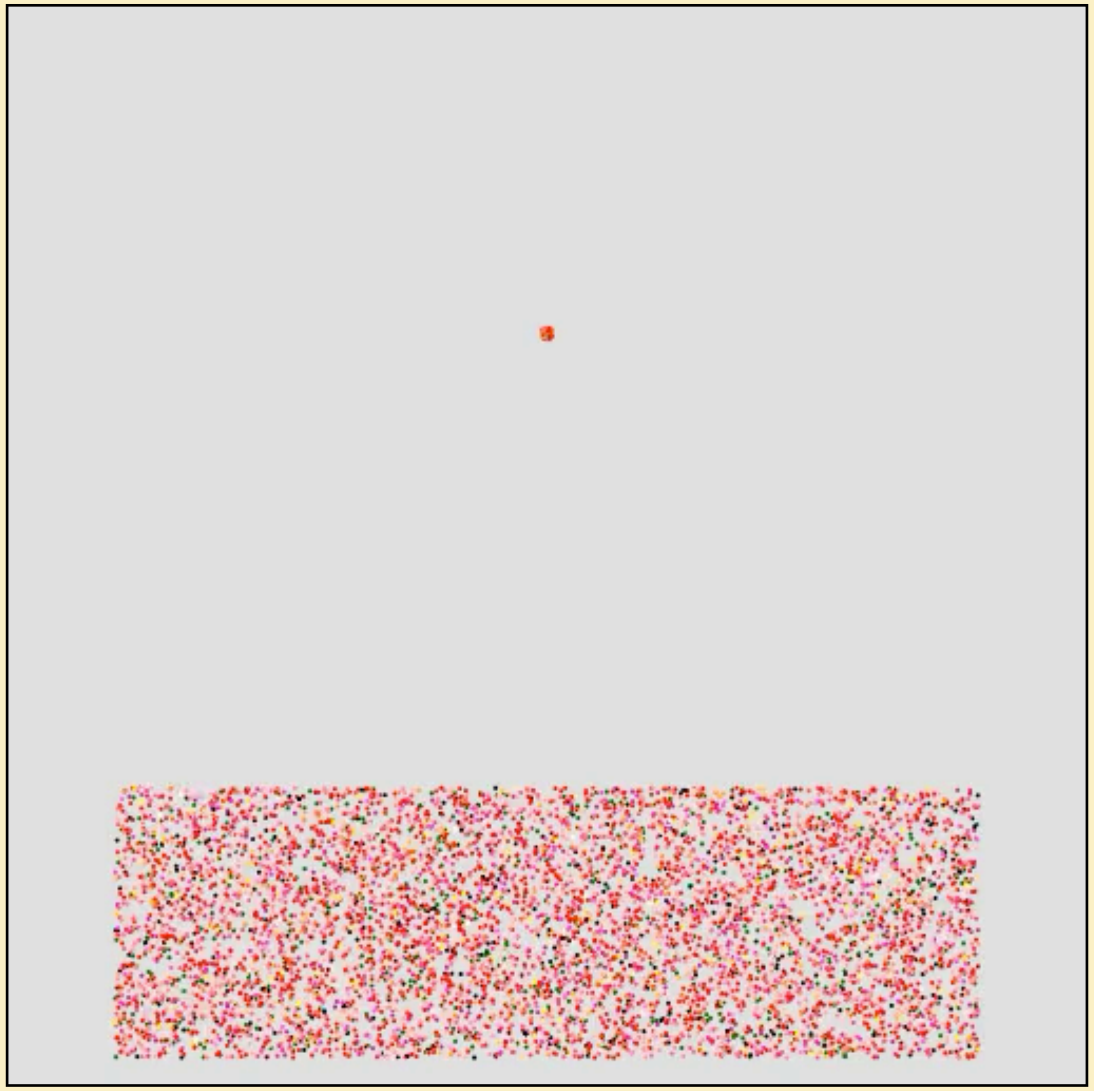
Progress: Part 2

Varying Material Density

Previously: Material Point density fixed

Now: material dependent! (Allows for heavier smaller objects)



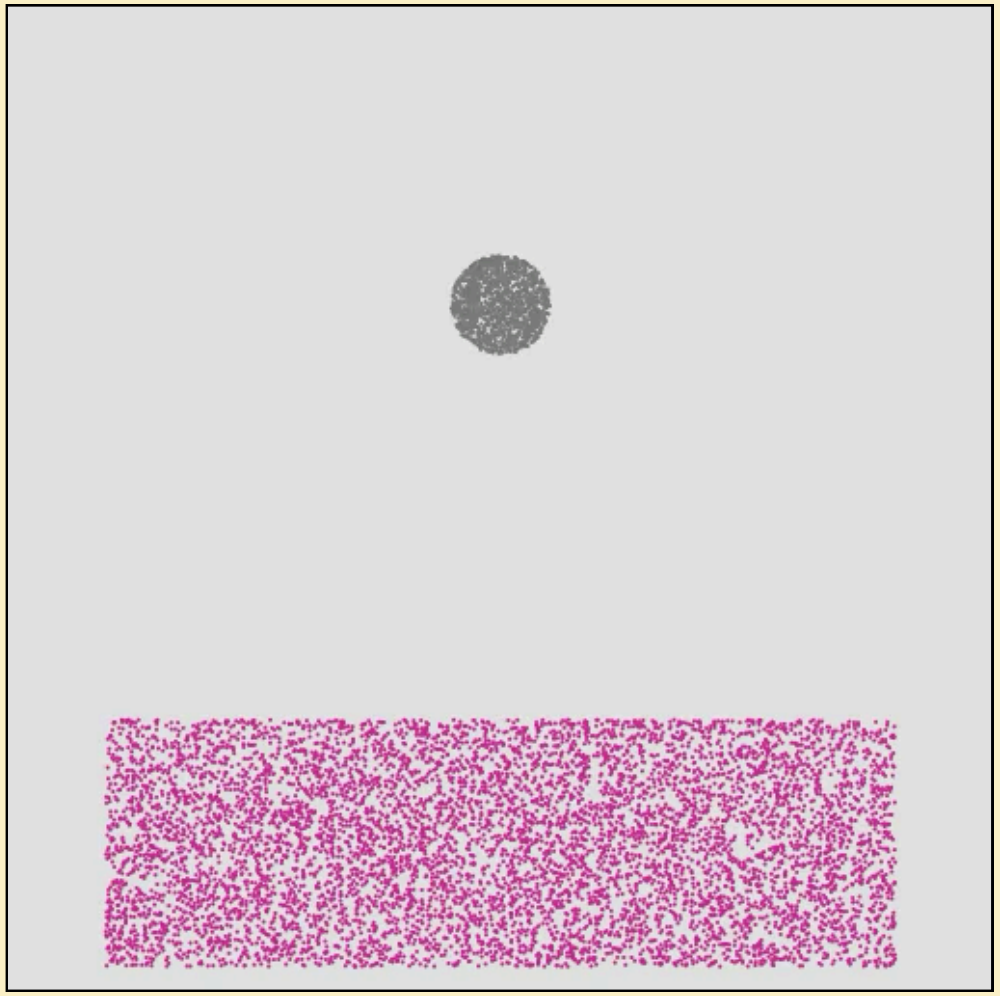


New Materials: Metal-Like

Must change **constitutive models** and parameters within **P2G step**

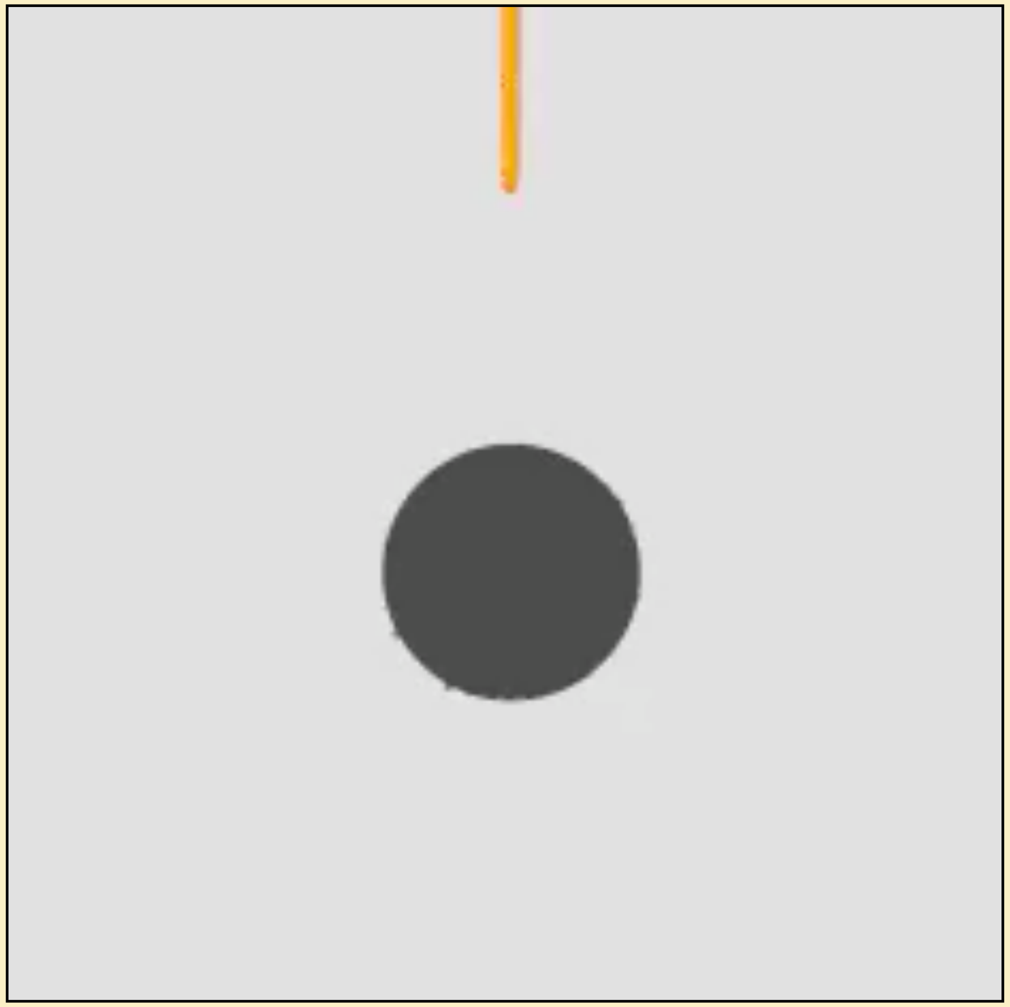
Jello: soft Neo-Hookean elastic material with a hardening factor ($h = 0.3$) applied to base Lamé parameters (μ_0, λ_0).

Metal-like: stiff Neo-Hookean elastic material, higher hardening factor ($h = 5.0$) for the base Lamé parameters.



New Materials: Honey

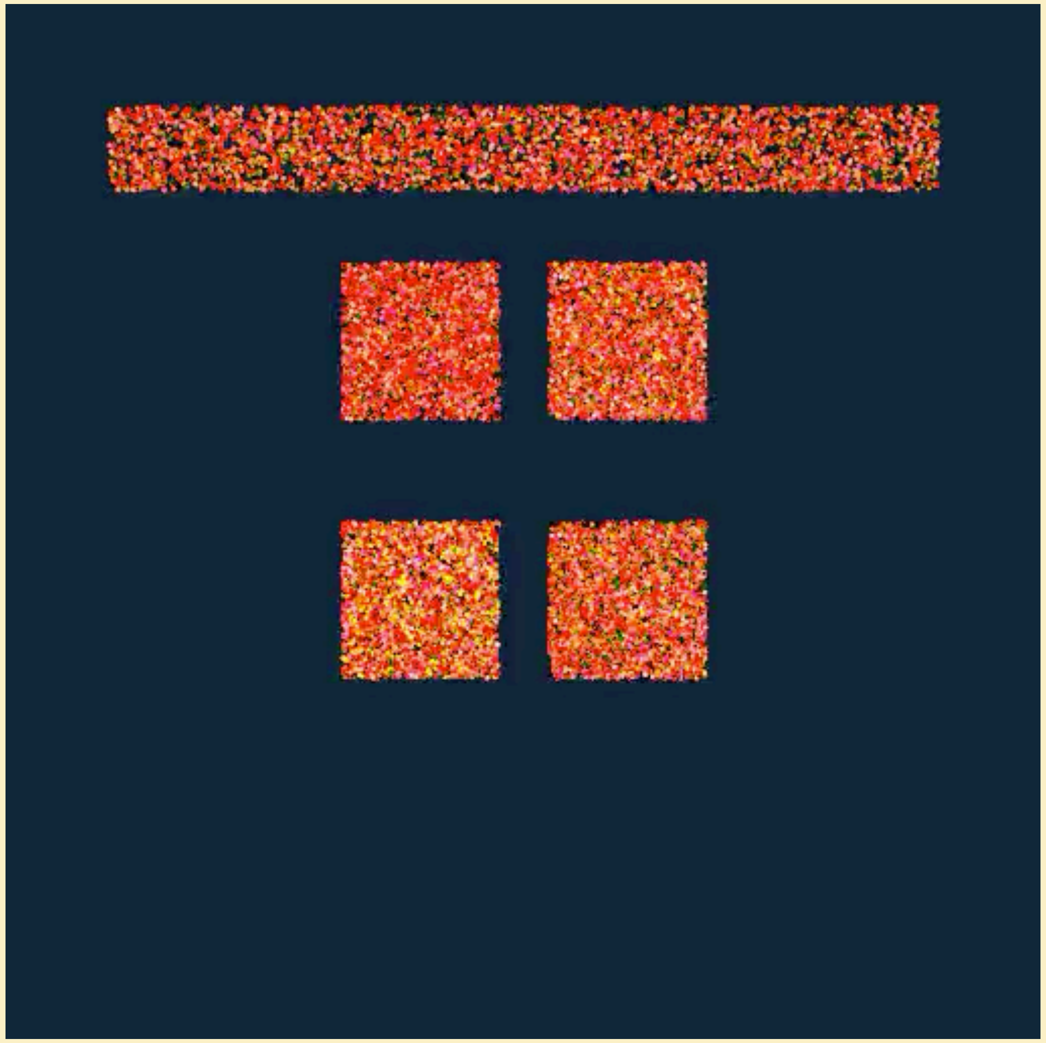
Honey: *high-viscosity* elastic material using large Lamé parameters; it flows (by resetting F_p to $I \cdot \sqrt{\det(F_p)}$) if its stress exceeds a defined *yield strength*, and includes explicit velocity damping in G2P.



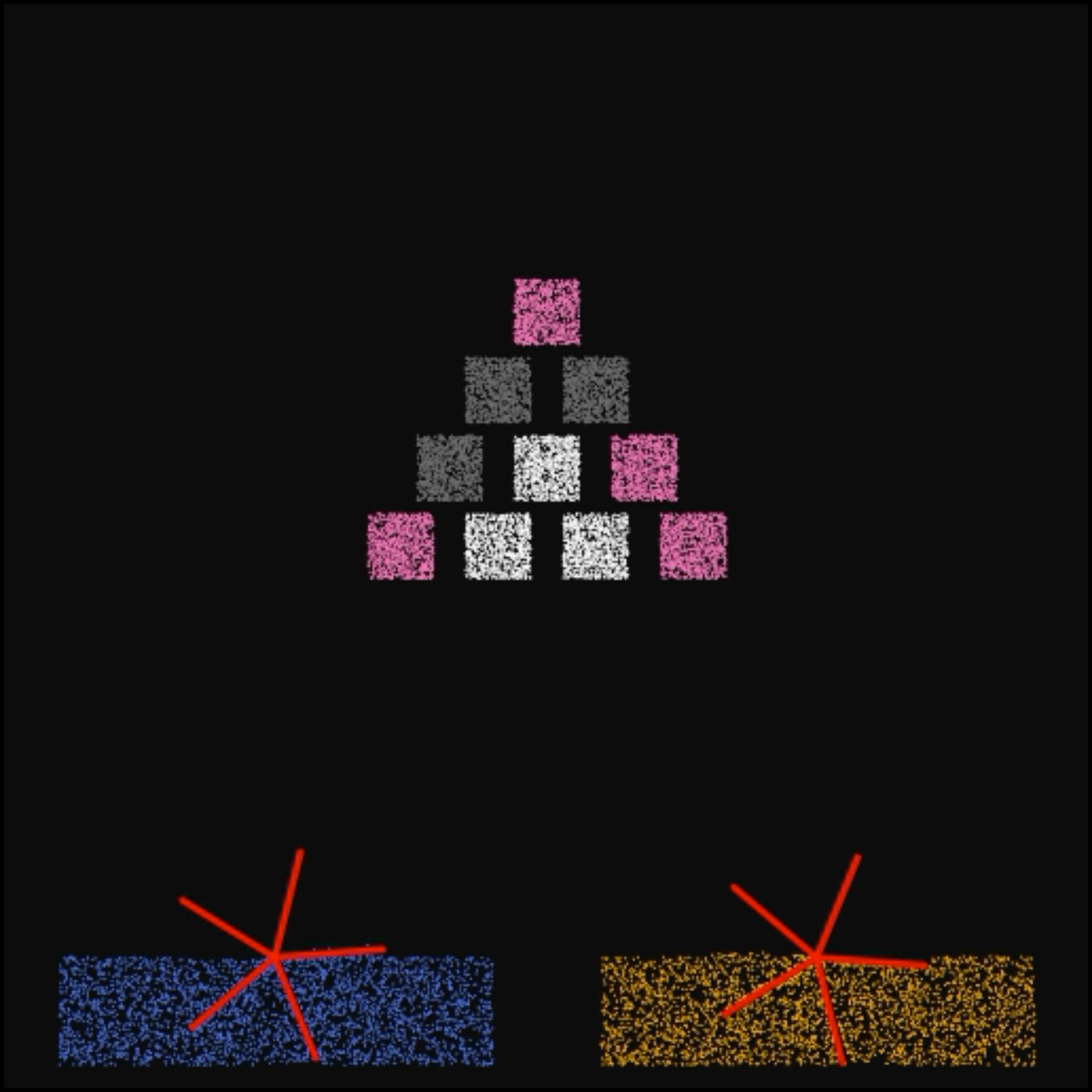
Multi-Material Interactions

Liquid: zero shear modulus ($\mu = 0$); its deformation gradient F_p was reset to remove shear and maintain stability.

Snow: elasto-plastic model w/ dynamic hardening factor ($h = f(J_p)$, J_p is plastic volume change) and plasticity enforced by clamping the singular values of F_p.



Multi-Material Interactions + Rigid Objects

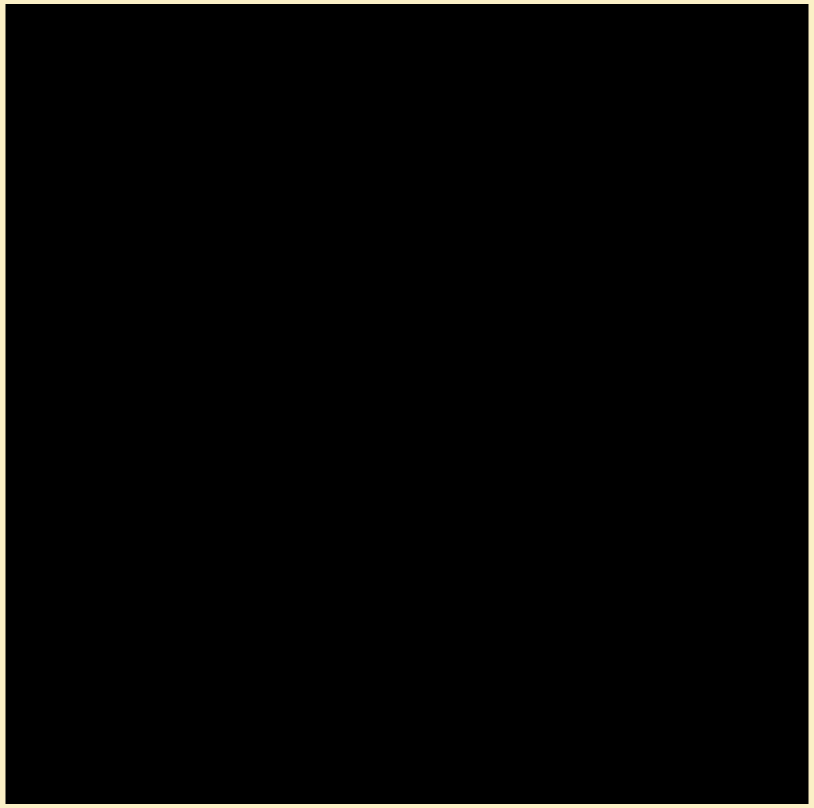


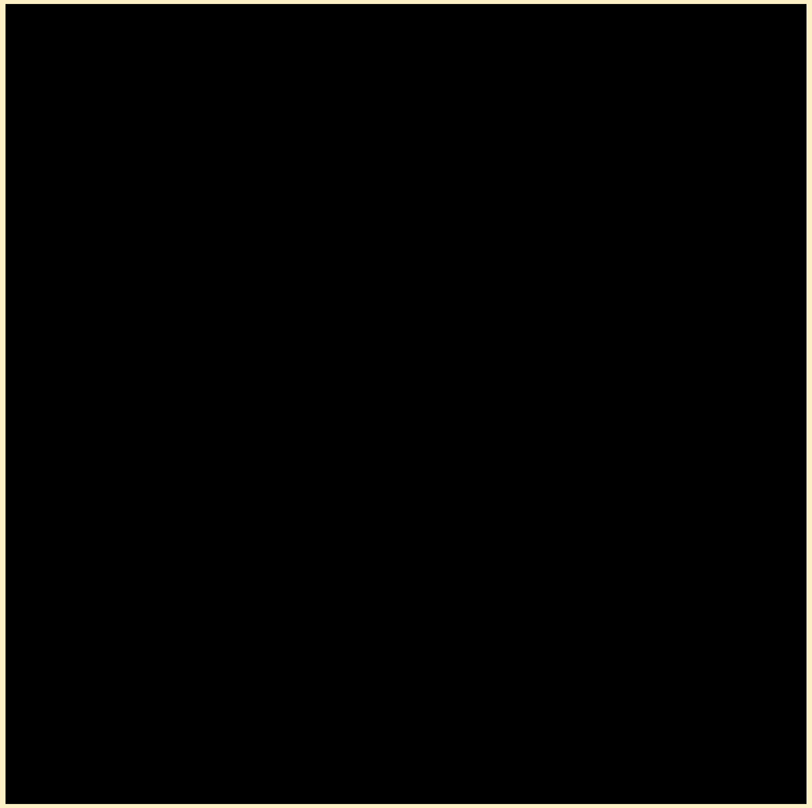
Progress: Part 3

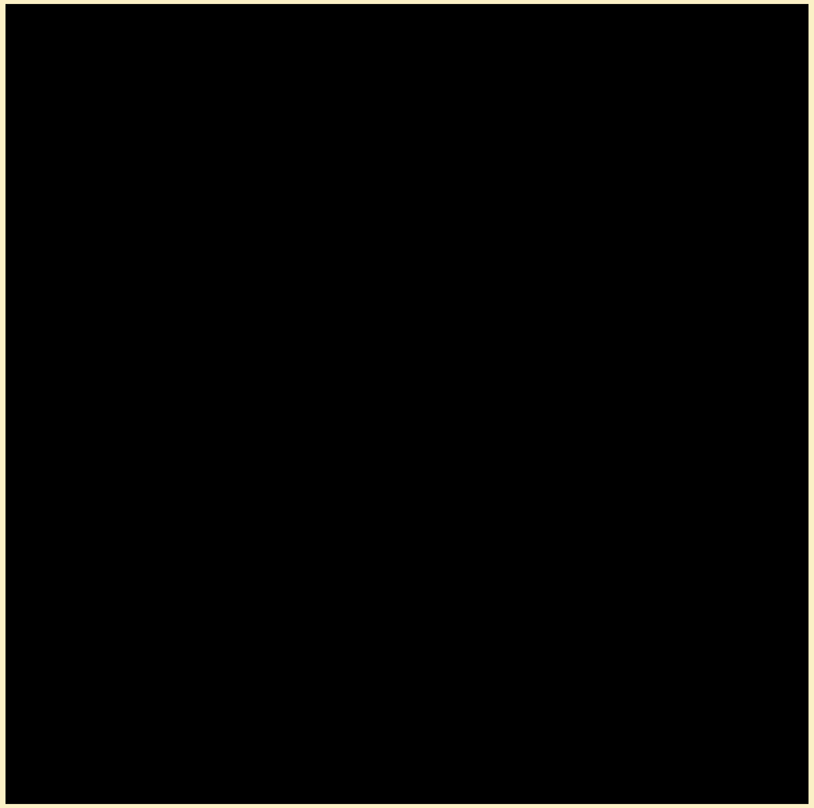
3D Water Simulation

Same Parameters With 2D Code

Different Container Sizes



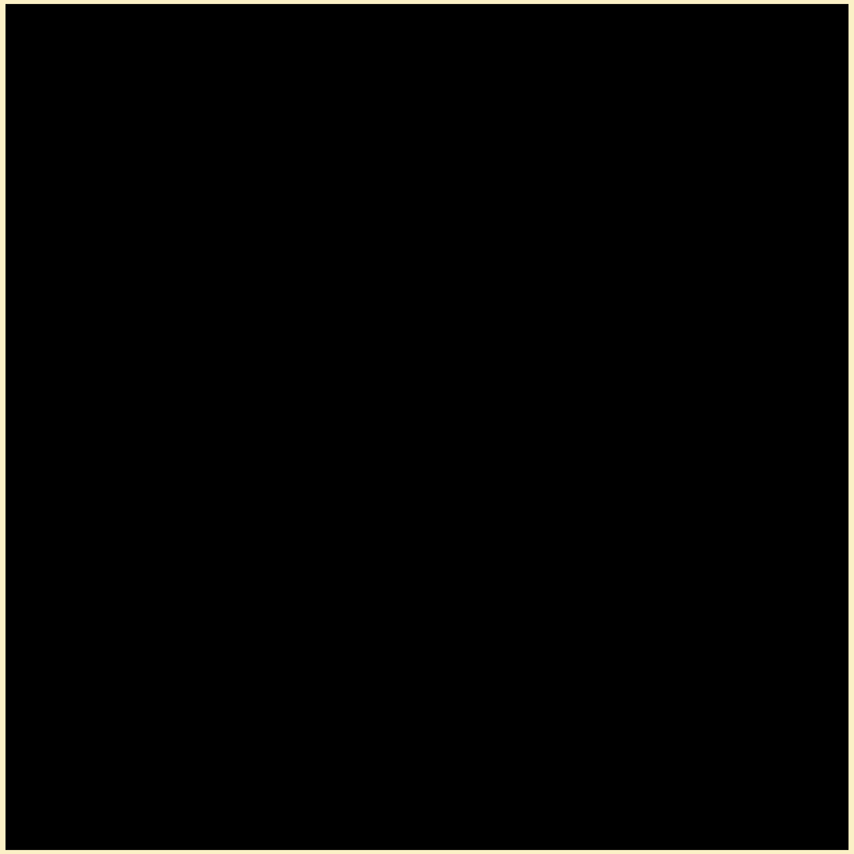




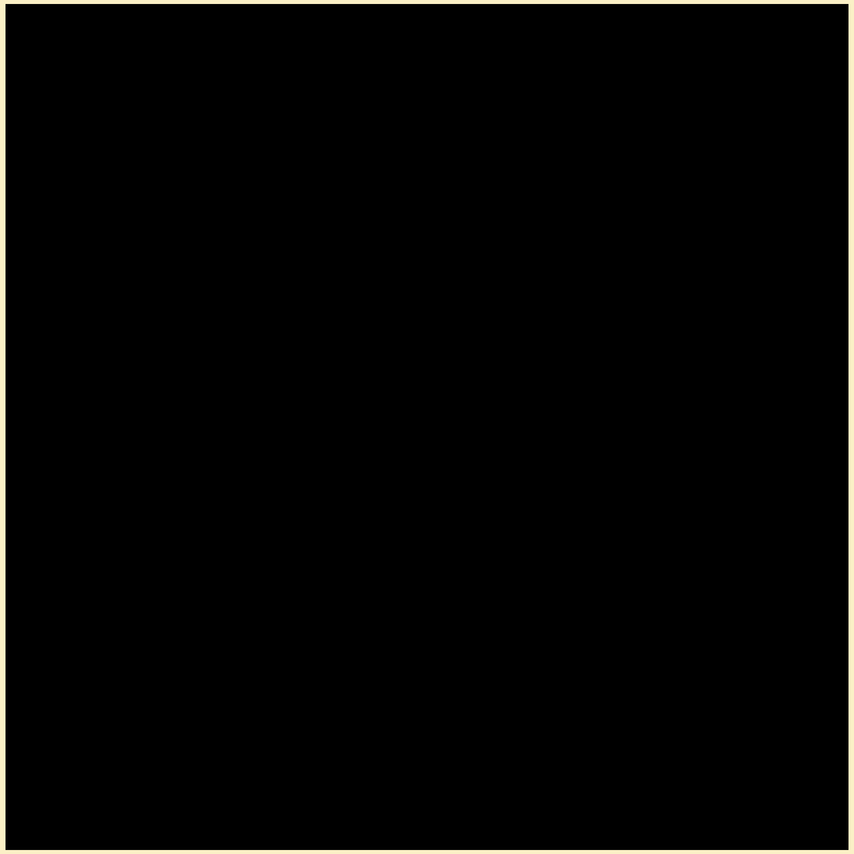
Ablation: Grid Resolution

Larger grid resolution means more accuracy but more computationally expensive.

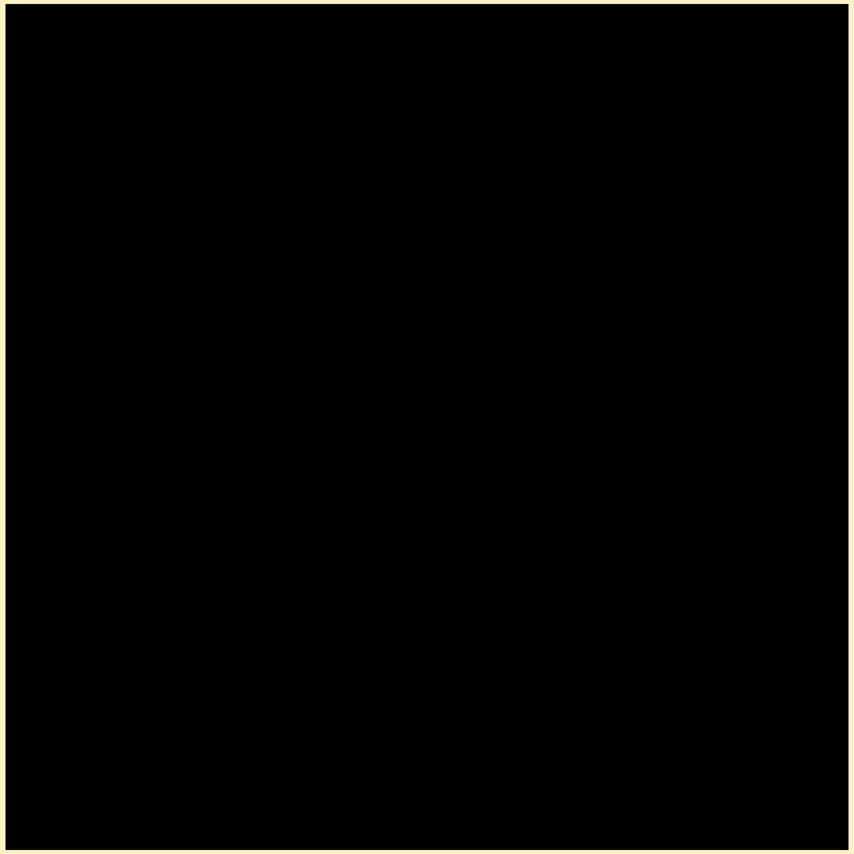
Can’t simulate, surface extract, and save OBJs at each frame on 100k particles :(



150



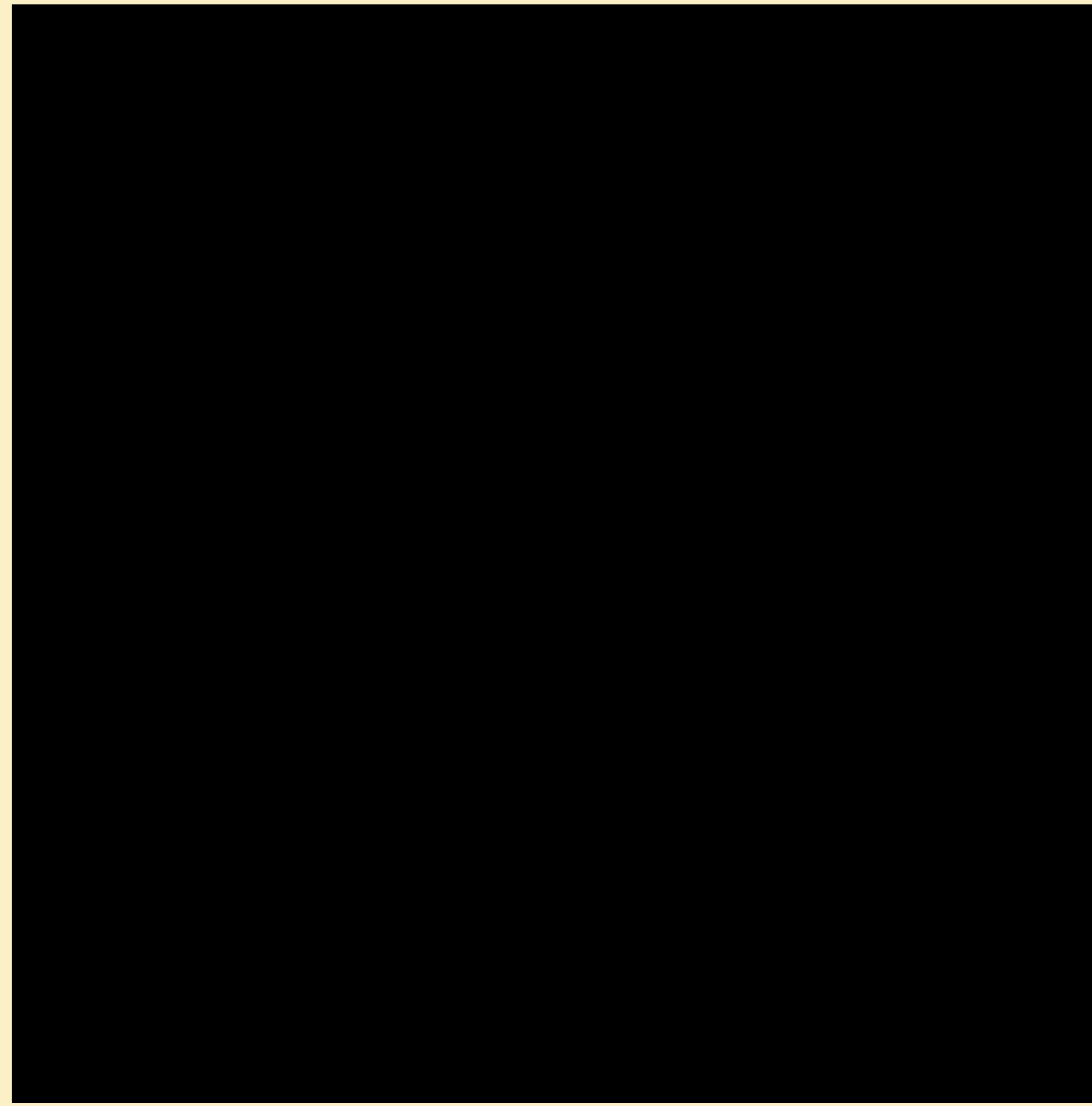
200



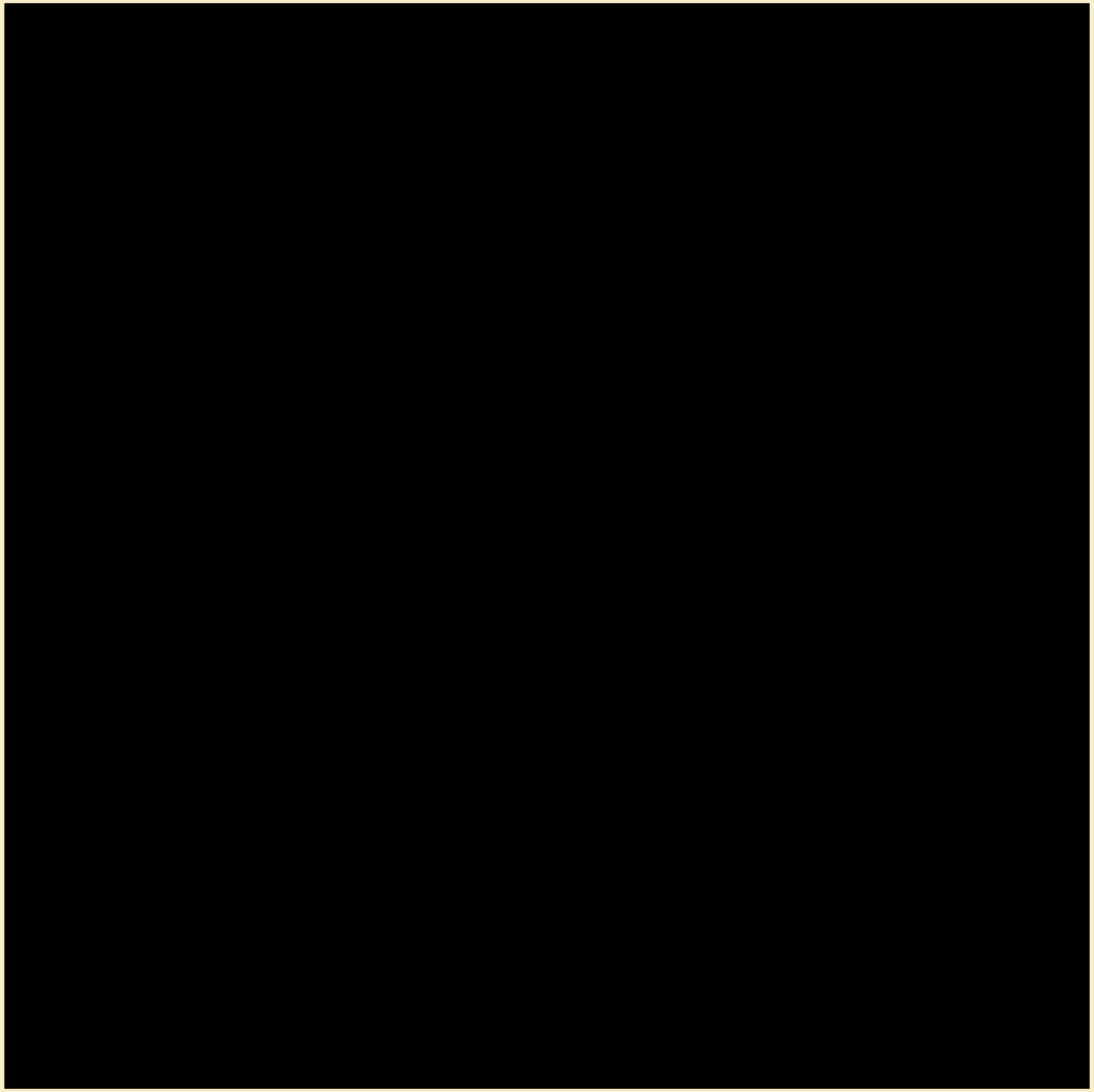
250

One-Way Rigid Body Interaction

1. Advect
2. Find closest point on the mesh to each particle
3. Detect Collision
4. Normal-based position correction and velocity reflection



Two Materials Collision



Stiff ball vs Snow Cube

Same parameters with 2D code is not accurate here

Progress: Part 4

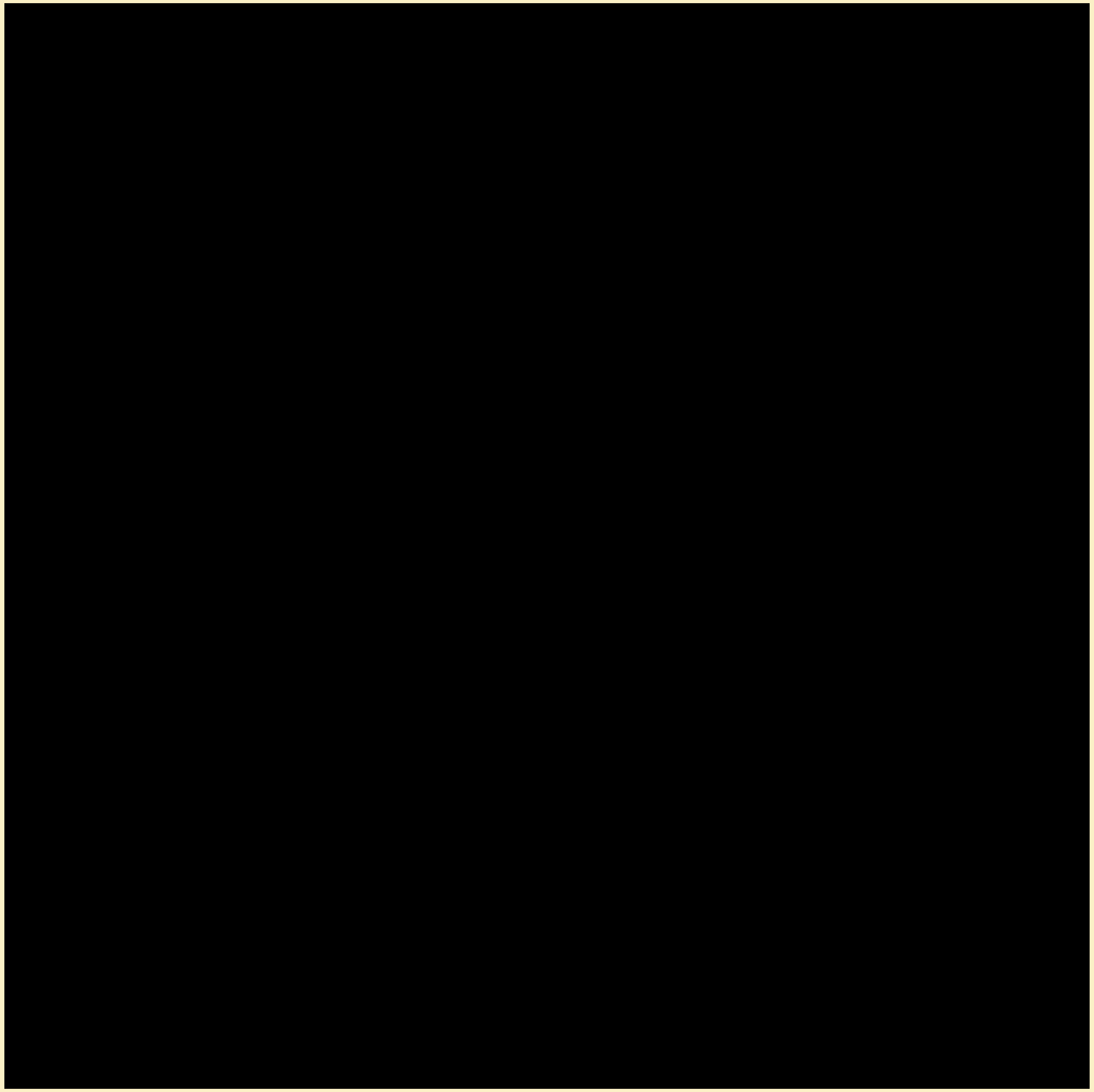
The Path to Blender

Simulate in Taichi, Render in Blender

Rendering Materials In Blender

1. Simulate material points in Taichi
2. Extract surfaces using marching cubes
3. Save mesh M_t at each timstep t
4. Load $\{M_t\}_{t=0}^{T}$ into Blender, assign materials
4. Render!

Surface Extraction: Marching Cubes



Taichi Visualization : /

Surface Extraction to Blender

Jello falling onto bunny

Blender Visualization! (╯°□°)╯︵ ┻━┻

Questions?