Position-Based Dynamics

Analysis and Implementation

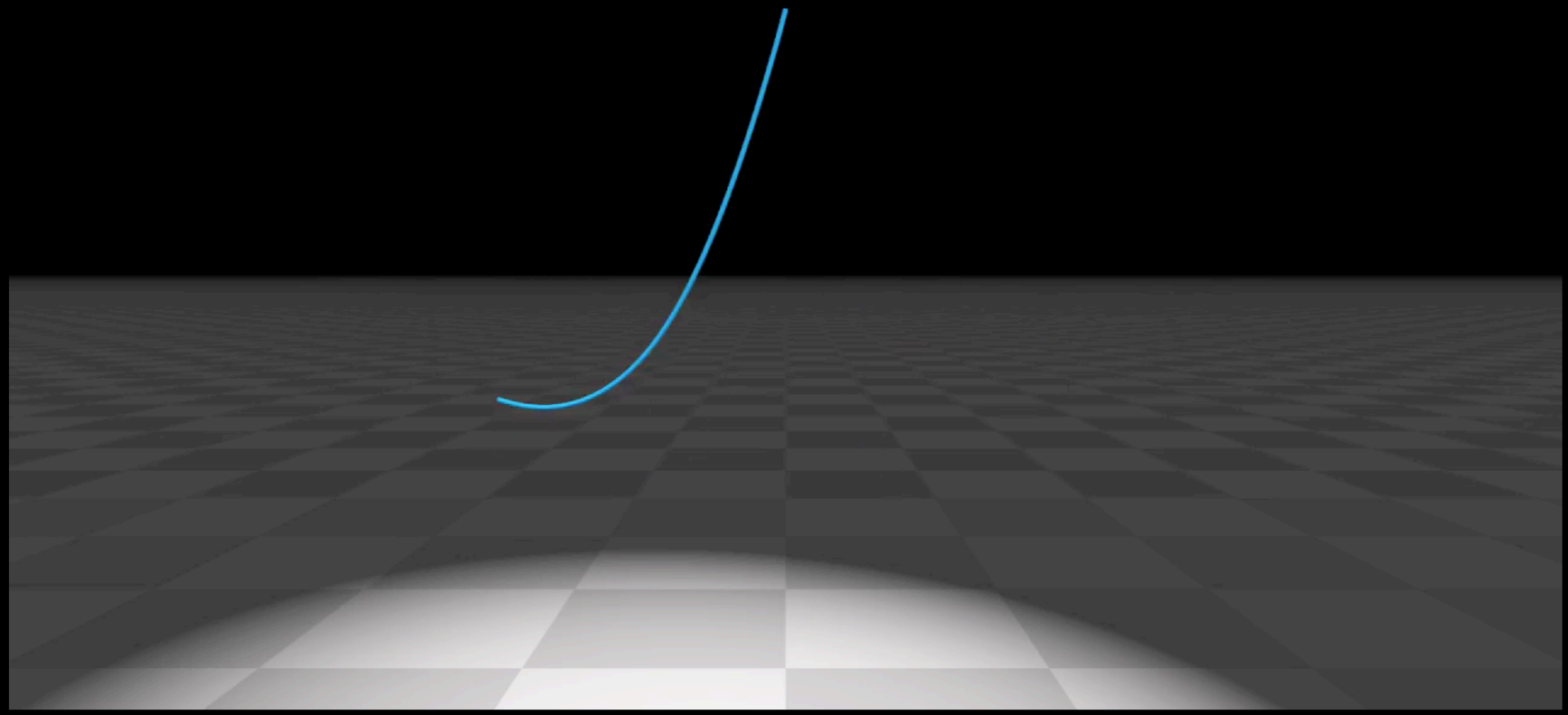
Miles Macklin



Analysis

Position-Based Dynamics

- Very stable
- Highly damped
- Example



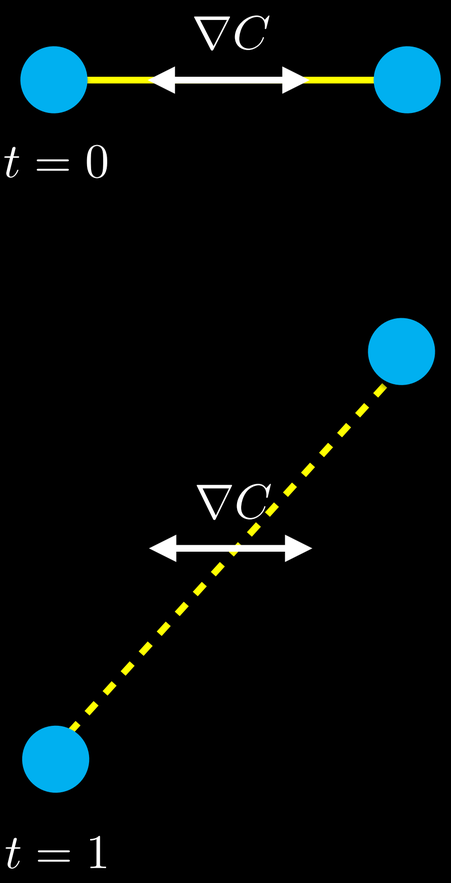
Verlet Integration

$$\mathbf{M}(\mathbf{x}^{n+1} - 2\mathbf{x}^{n} + \mathbf{x}^{n-1}) = \Delta t^2 \mathbf{f}_{ext}(\mathbf{x}^{n})$$

- Symplectic (long term energy conservation)
- So why the damping?
- Where are the constraint forces?

Constrained Verlet Integration

- Popular in other fields, e.g.: Molecular Dynamics
- e.g. SHAKE / RATTLE
- Differs from PBD in that constraint directions are fixed, and calculated at the start of the time-step (c.f. figure).
- Fixing constraint directions can lead to instability



Implicit Euler Integration

$$\mathbf{v}^{n+1} = \mathbf{v}_n + \Delta t \mathbf{M}^{-1} \mathbf{f}(\mathbf{x}^{n+1})$$
$$\mathbf{x}^{n+1} = \mathbf{x}_n + \Delta t \mathbf{v}^{n+1}$$

- Forces evaluated at end of the time-step
- Equivalent to:

$$\mathbf{M}(\mathbf{x}^{n+1} - 2\mathbf{x}^n + \mathbf{x}^{n-1}) = \Delta t^2 \mathbf{f}(\mathbf{x}^{n+1})$$

- Very stable
- Highly damped

Implicit Euler as an optimization problem

$$\mathbf{M}(\mathbf{x}^{n+1} - 2\mathbf{x}^n + \mathbf{x}^{n-1}) = \Delta t^2 \mathbf{f}(\mathbf{x}^{n+1})$$

- Can be seen as the first order optimality conditions for the following minimization:

$$\min_{\mathbf{x}^{n+1}} \quad \frac{1}{2}(\mathbf{x}^{n+1} - \tilde{\mathbf{x}})^T \mathbf{M}(\mathbf{x}^{n+1} - \tilde{\mathbf{x}}) + \Delta t^2 E(\mathbf{x}^{n+1})$$

- Predicted (inertial) position:

$$\begin{aligned}\tilde{\mathbf{x}} &= 2\mathbf{x}^n - \mathbf{x}^{n-1} + \mathbf{M}^{-1}\mathbf{f}_{ext} \\ &= \mathbf{x}^n + \Delta t \mathbf{v}^n + \mathbf{M}^{-1}\mathbf{f}_{ext}\end{aligned}$$

- See [Goldenthal 2007]

Implicit Euler as an optimization problem

$$\min_{\mathbf{x}^{n+1}} \quad \frac{1}{2}(\mathbf{x}^{n+1} - \tilde{\mathbf{x}})^T \mathbf{M}(\mathbf{x}^{n+1} - \tilde{\mathbf{x}}) + \Delta t E(\mathbf{x}^{n+1})$$

- As $E \to \infty$
- We obtain the following constrained minimization:

$$\begin{aligned} \min_{\mathbf{x}^{n+1}} \quad & \frac{1}{2}(\mathbf{x}^{n+1} - \tilde{\mathbf{x}})^T \mathbf{M}(\mathbf{x}^{n+1} - \tilde{\mathbf{x}}) \\ \text{s.t.} \quad & C_i(\mathbf{x}^{n+1}) = 0 \end{aligned}$$

- Searching for the point closest to the predicted (inertial) position that lies on the constraint manifold

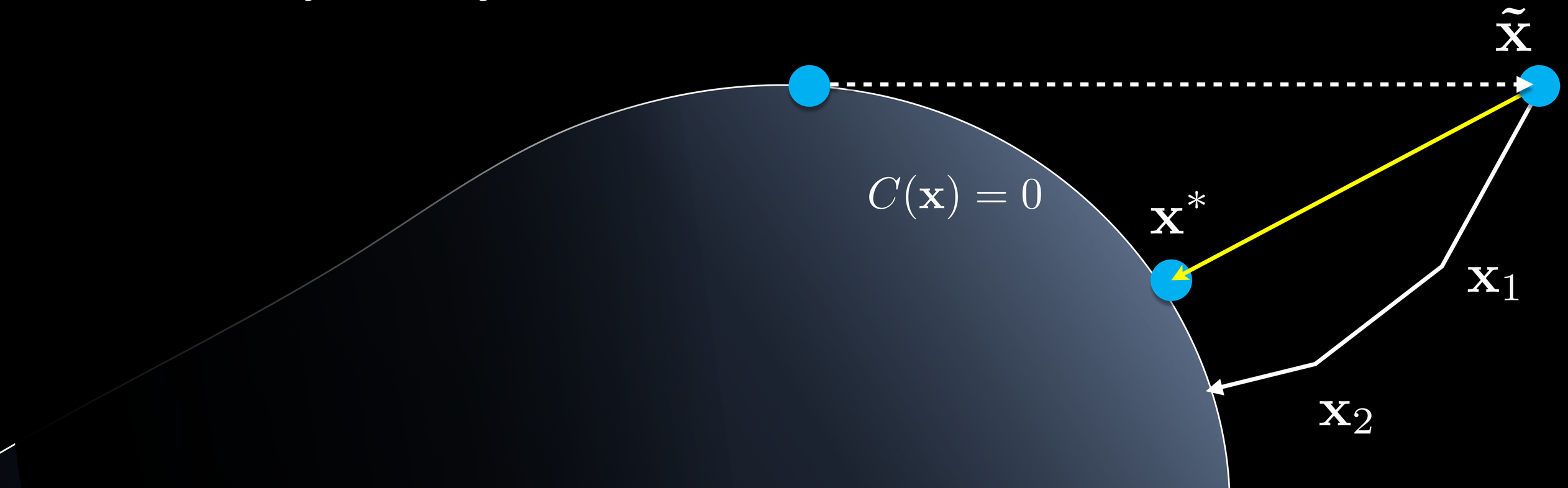
Implicit Euler as an optimization problem

$$\begin{aligned} \min_{\mathbf{x}^{n+1}} \quad & \frac{1}{2}(\mathbf{x}^{n+1} - \tilde{\mathbf{x}})^T \mathbf{M}(\mathbf{x}^{n+1} - \tilde{\mathbf{x}}) \\ \text{s.t.} \quad & C_i(\mathbf{x}^{n+1}) = 0 \end{aligned}$$

- Minimization gives the following algorithm:
  1. Predict new position $\tilde{\mathbf{x}}$
  2. Project to C(x) = 0
- General solver would require Newton's method
- PBD solves approximately using modified 'Fast Projection'

PBD and Implicit Euler

- First iteration of PBD is equivalent to the first iteration of backward Euler Newton solve $(\mathbf{x_0} = \tilde{\mathbf{x}})$
- See Projective Dynamics [Bouaziz et al. 14]



Second Order Implicit Euler

- First order backward Euler (BDF1)

$$\mathbf{v}^{n+1} = \mathbf{v}_n + \Delta t \mathbf{M}^{-1} \mathbf{f}(\mathbf{x}^{n+1})$$
$$\mathbf{x}^{n+1} = \mathbf{x}_n + \Delta t \mathbf{v}^{n+1}$$

- Second order backward Euler (BDF2)

$$\mathbf{v}^{n+1} = \frac{4}{3}\mathbf{v}^n - \frac{1}{3}\mathbf{v}^{n-1} + \frac{2}{3}\Delta t \mathbf{M}^{-1} \mathbf{f}(\mathbf{x}^{n+1})$$
$$\mathbf{x}^{n+1} = \frac{4}{3}\mathbf{x}^n - \frac{1}{3}\mathbf{x}^{n-1} + \frac{2}{3}\Delta t \mathbf{v}^{n+1}$$

Second Order PBD

- First order prediction

$$\tilde{\mathbf{x}} = \mathbf{x}^n + \Delta t \mathbf{v}^n + \Delta t^2 \mathbf{M}^{-1} \mathbf{f}_{ext}$$

- Second order prediction

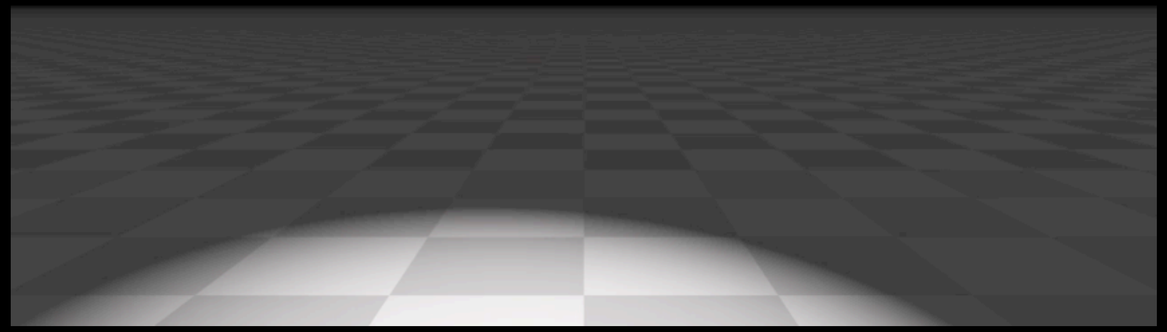
$$\tilde{\mathbf{x}} = \frac{4}{3}\mathbf{x}^n - \frac{1}{3}\mathbf{x}^{n-1} + \frac{8}{9}\Delta t \mathbf{v}^n - \frac{2}{9}\Delta t \mathbf{v}^{n-1} + \frac{4}{9}\Delta t^2 \mathbf{M}^{-1} \mathbf{f}_{ext}$$

- Update velocity using second order backward difference

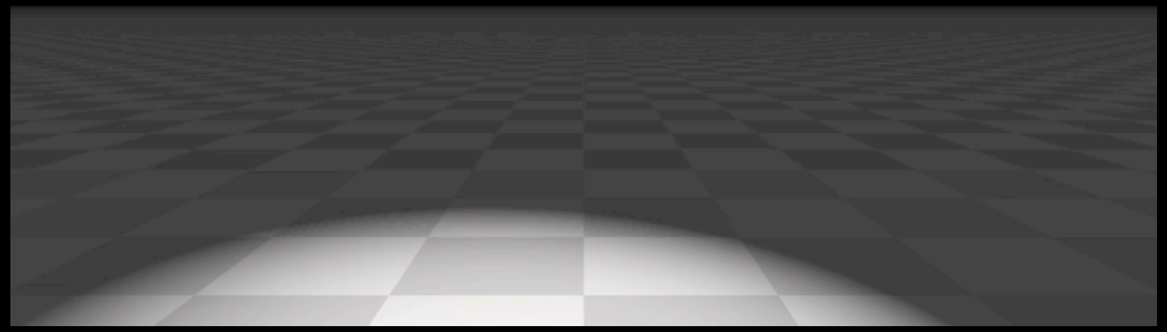
$$\mathbf{v}^{n+1} = \frac{1}{\Delta t}\left[\frac{3}{2}\mathbf{x}^{n+1} - 2\mathbf{x}^n + \frac{1}{2}\mathbf{x}^{n-1}\right].$$

- See [English 08]

Second Order PBD

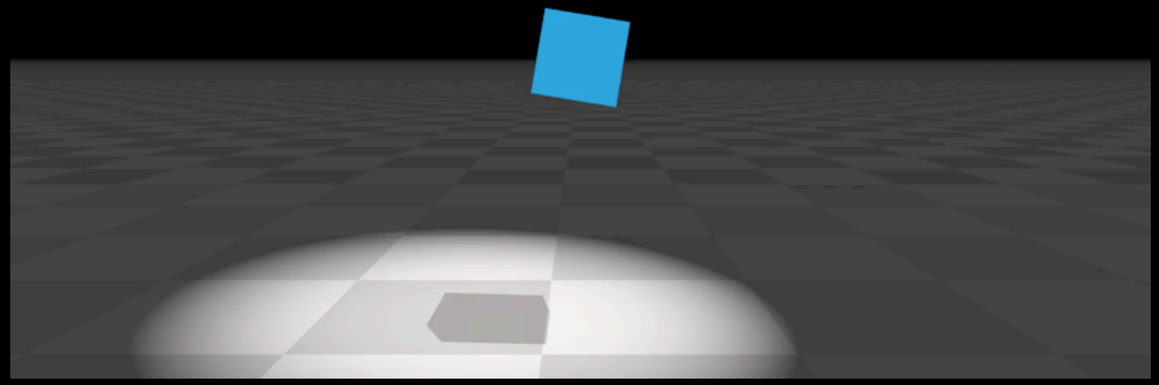


First Order

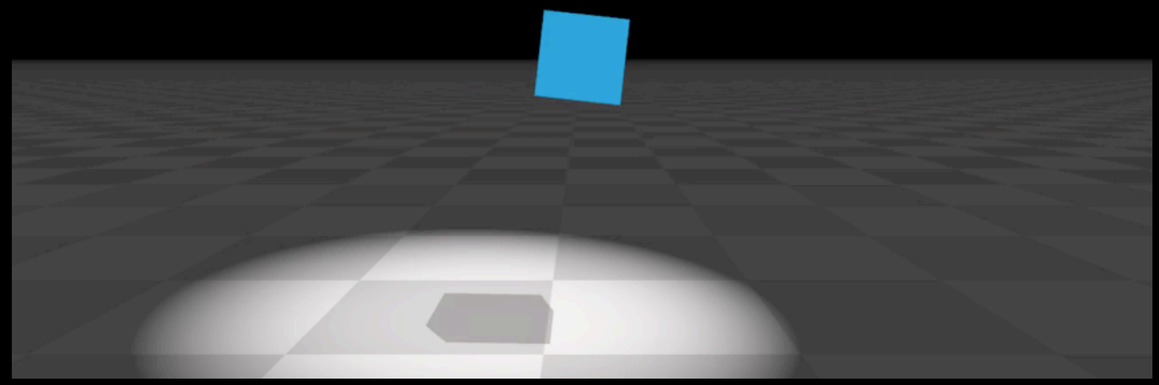


Second Order

Second Order PBD

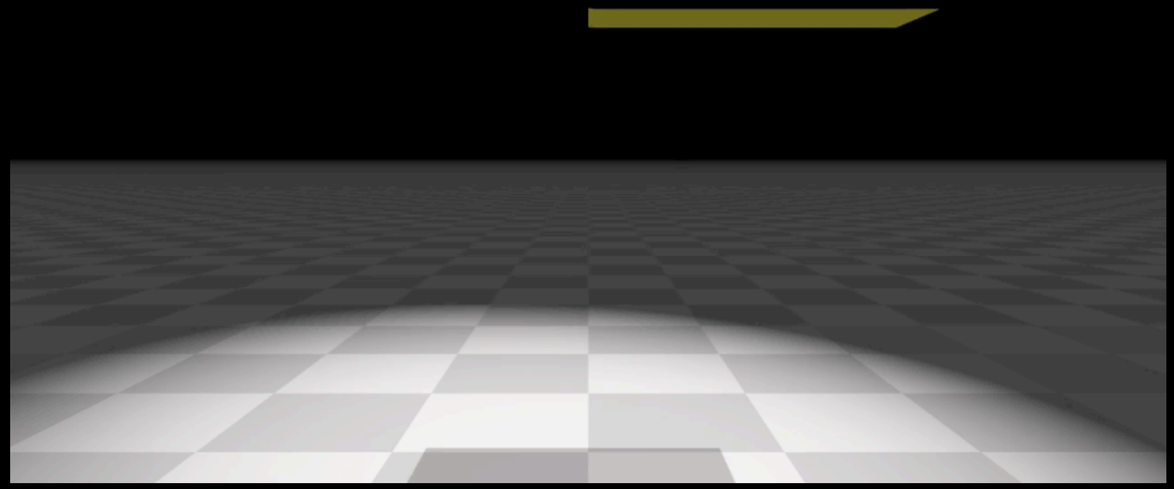


First Order

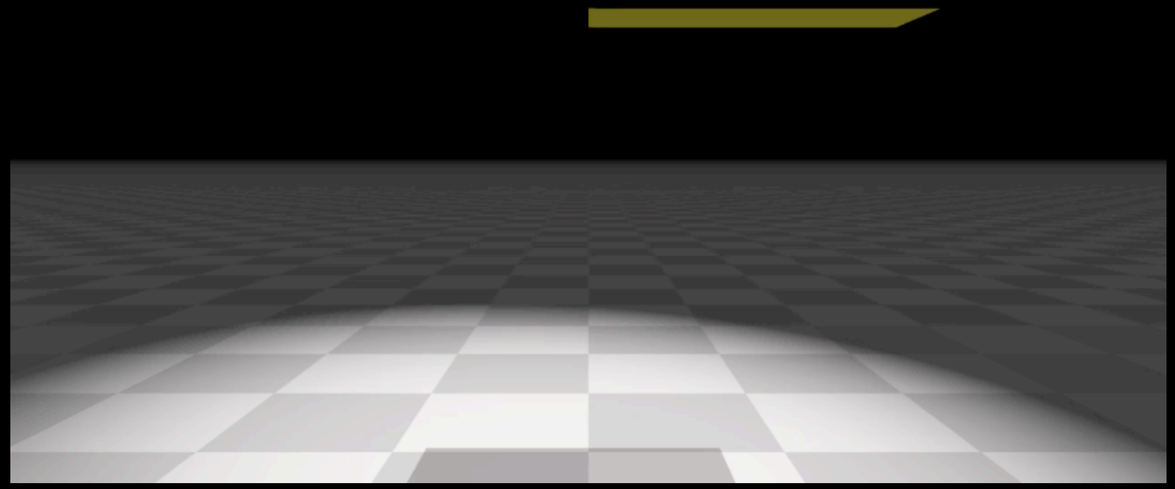


Second Order

Second Order PBD



First Order



Second Order

Second Order PBD

- Significantly less damping
- Positions stay closer to constraint manifold
- Requires fewer constraint iterations!
- Non-smooth events (contact) need special handling

More Close Relatives

- Nucleus [Stam]
- Strain Limiting [Provot]
- Viscoelastic Particles [Clavet]

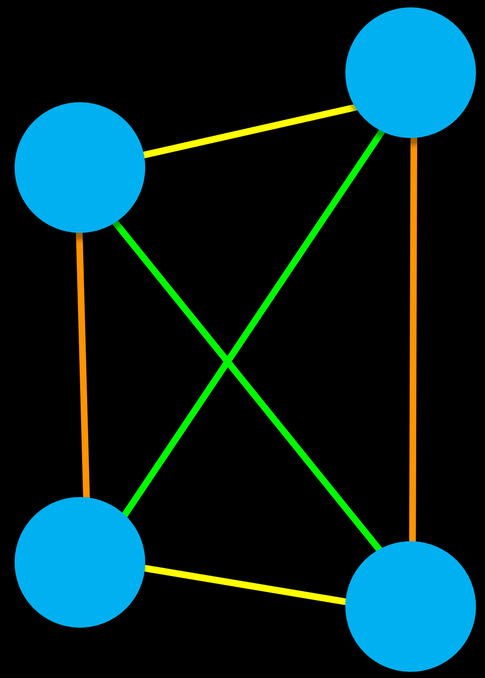
Implementation

Parallel PBD

- Gauss-Seidel inherently serial
- Parallel options:
  - Graph Coloring methods
  - Jacobi methods
  - Hybrid methods

Graph Coloring Methods

- Break constraint graph into independent sets
- Solve the constraints in a set in parallel
- “Batched” Gauss-Seidel
- Requires synchronization between each set
- Size of sets decreases -> poor utilisation



3 Color Graph

Jacobi Methods

- Process each constraint individually
- Sum up contributions on each particle

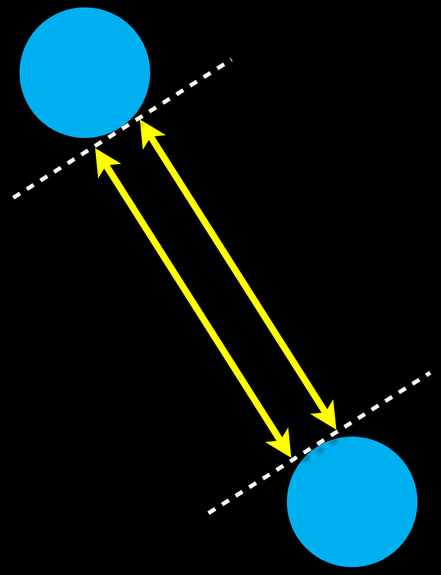
Particle-centric approach (gather)	Constraint-centric approach (scatter)

`foreach particle (in parallel)`
`{`
`  foreach constraint`
`  {`
`  calculate constraint error`
`  update delta`
`  }`
`}`
	`foreach constraint (in parallel)`
	`{`
	`  calculate constraint error`
	`  foreach particle`
	`  {`
	`  update delta (atomically)`
	`  }`
	`}`

Jacobi Methods

- Problem: system matrix can be indefinite, Jacobi will not converge, e.g.: for redundant constraints (cf. figure)
- Regularized Jacobi iteration via averaging [Bridson et al. 02]
- Sum all constraint deltas together and divide by constraint count for that particle

$$\mathbf{x_i} \leftarrow \mathbf{x_i} + \frac{1}{n_i} \sum_{n_i} \lambda_j \nabla C_j$$



- Successive-over relaxation by user parameter omega [0,2]:

$$\mathbf{x_i} \leftarrow \mathbf{x_i} + \frac{\omega}{n_i} \sum_{n_i} \lambda_j \nabla C_j$$

Comparison

Method	Advantages	Disadvantages
Batched Gauss-Seidel	Good Convergence Very Robust	Graph Coloring Synchronization
Jacobi	Trivial Parallelism	Slow Convergence Less Robust

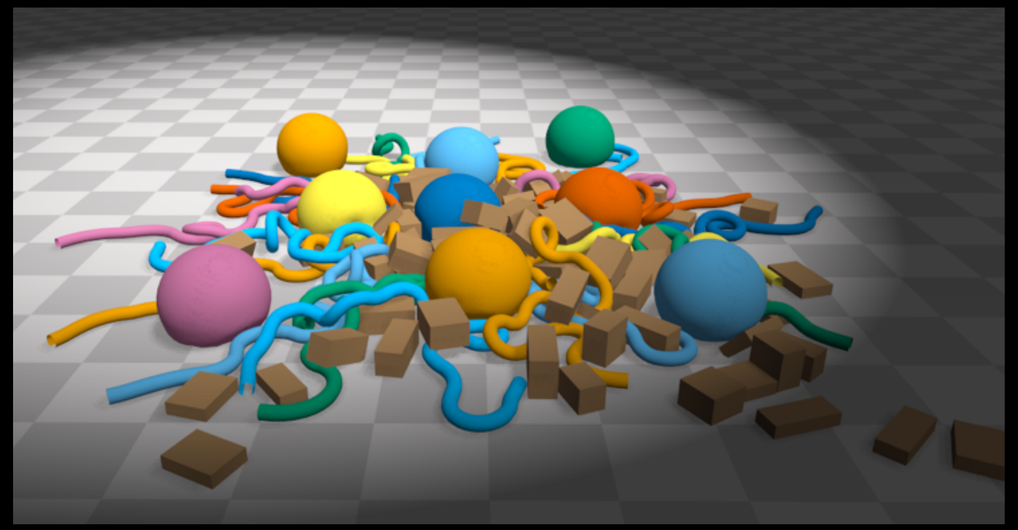
Hybrid Methods

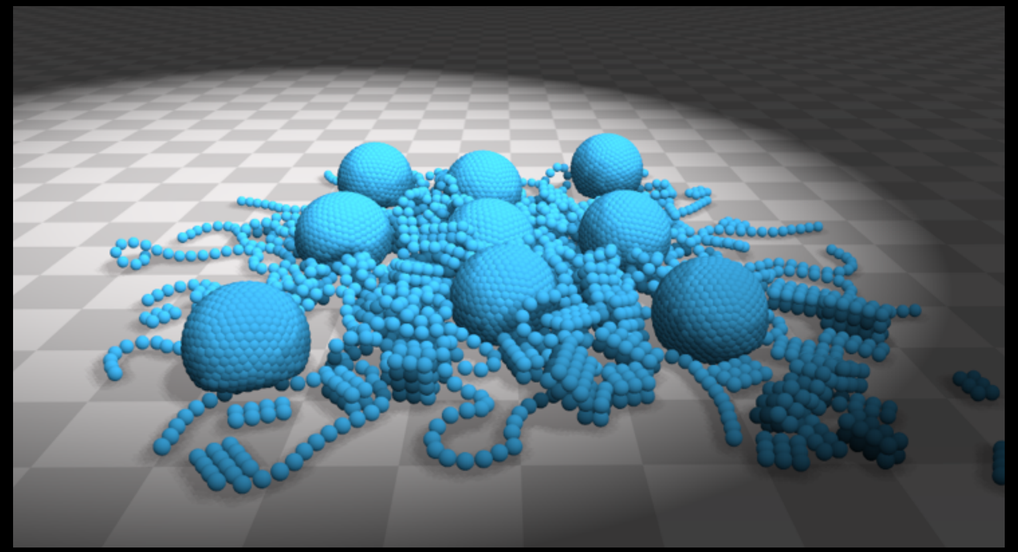
- Best of both worlds!
- Perform graph-coloring
- Upper limit on number of colors
- Process everything else with Jacobi

Unified Solver

Everything is a set of particles connected by constraints

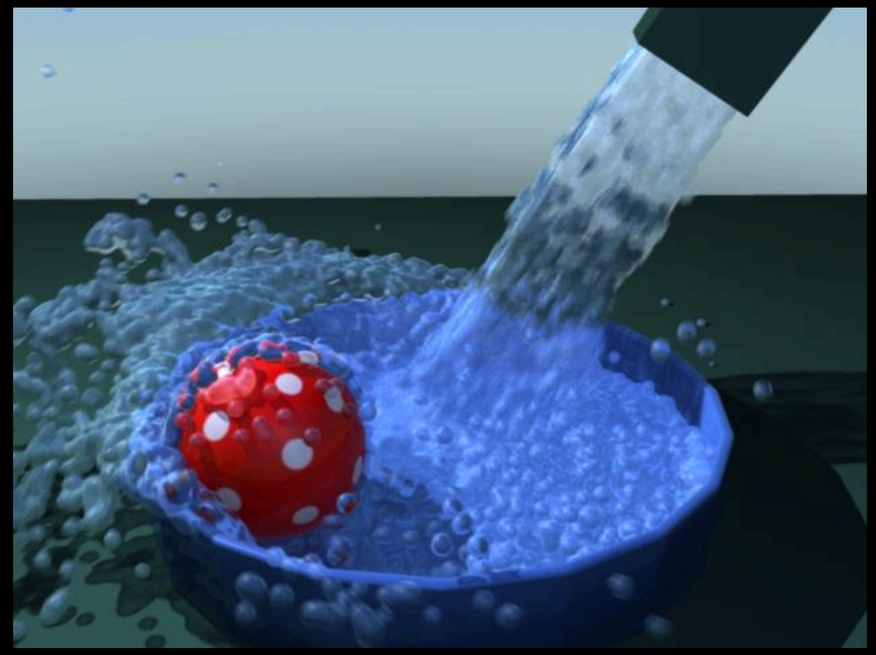
- Simplifies collision detection
- Two-way interaction of all object types:
  - Cloth
  - Deformables
  - Fluids
  - Rigid Bodies
- Fits well on the GPU





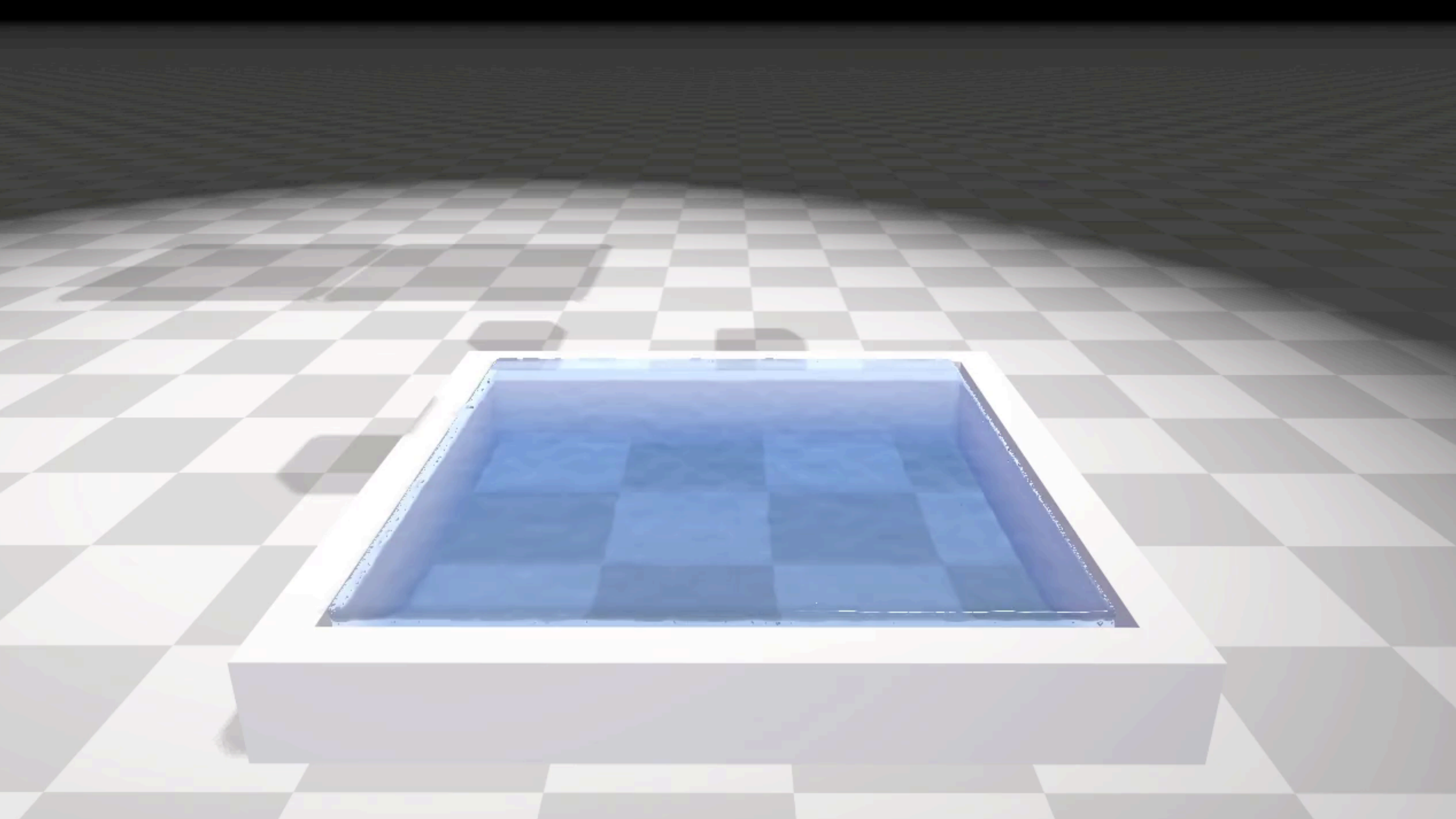
Related Work

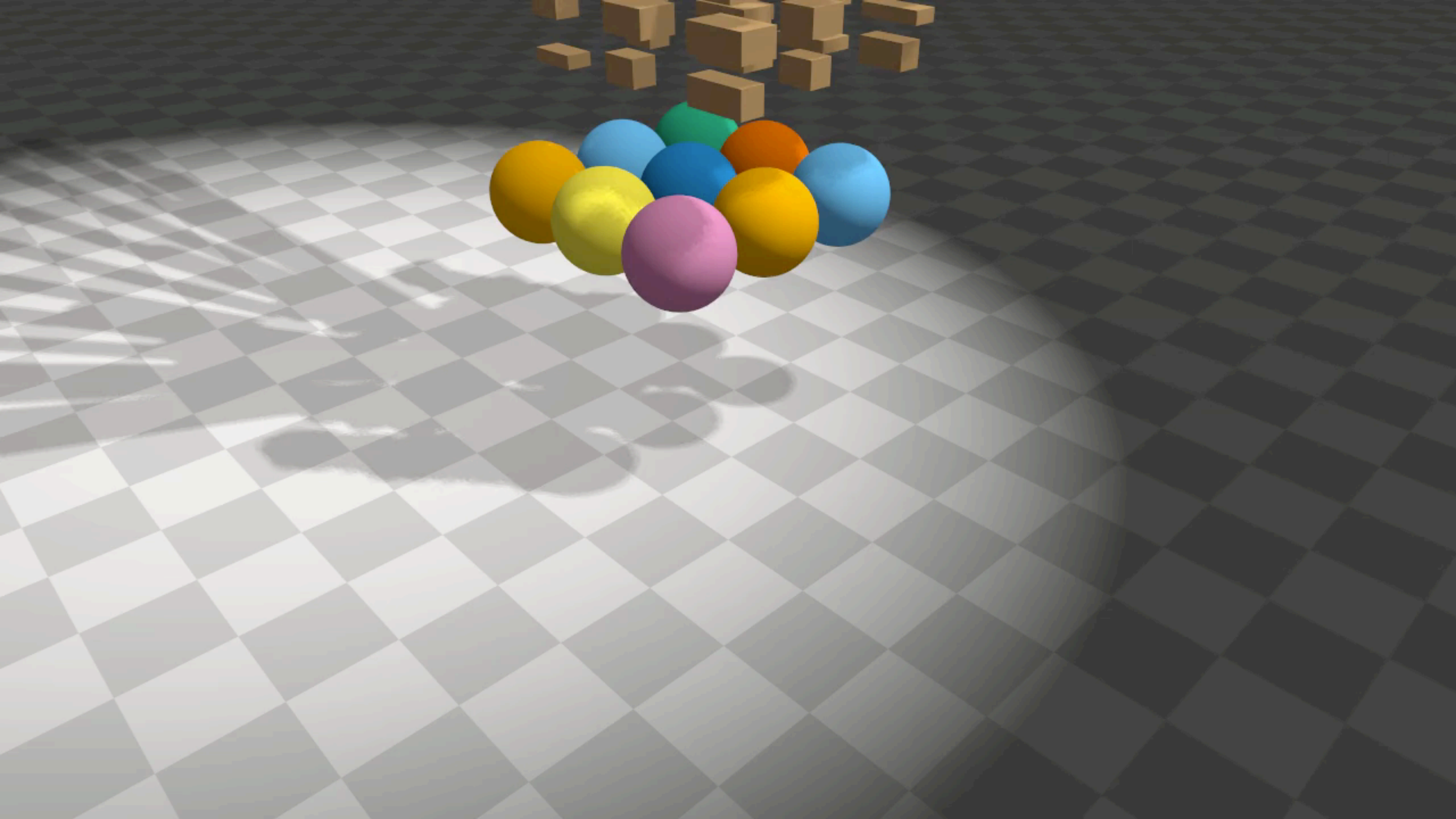
- Unified solvers popular in offline visual effects, e.g.:
  - Maya’s Nucleus solver (nCloth, nParticles) [Stam09]
  - Softimage's Lagoa (fluids, elastics, granular materials)
- Goal: recreate these packages in real-time

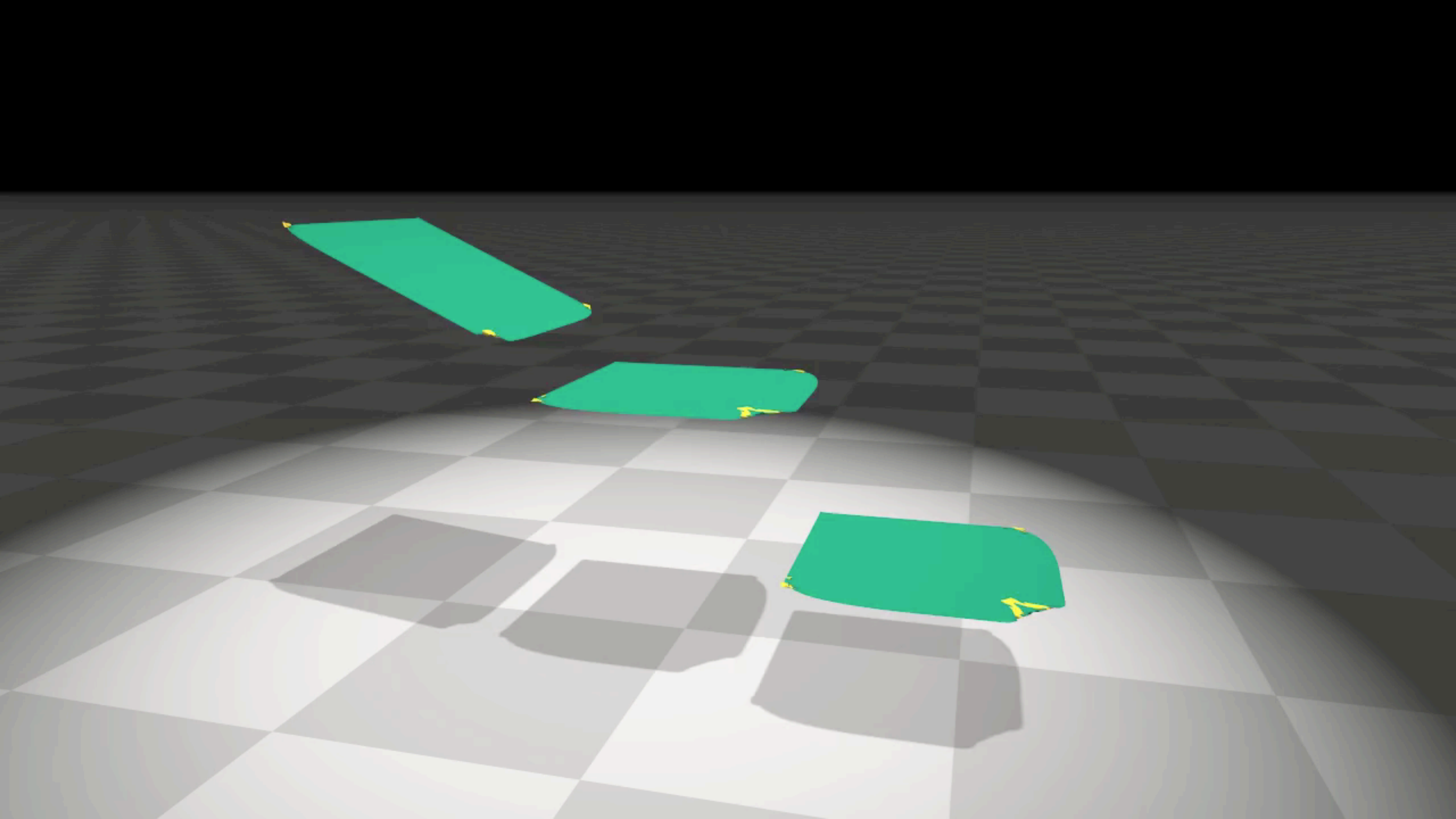


Maya nDynamics

Examples



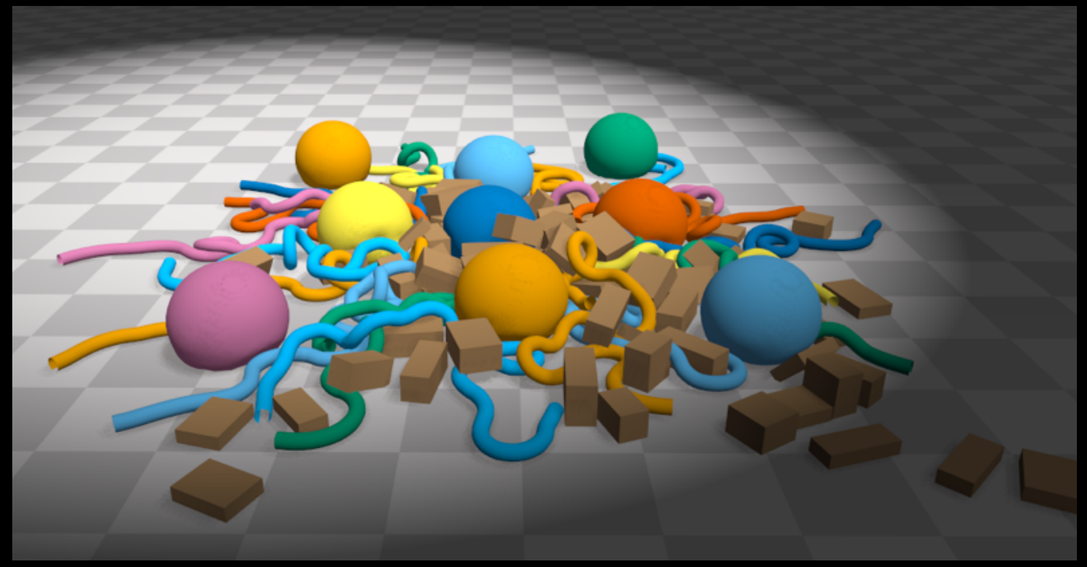


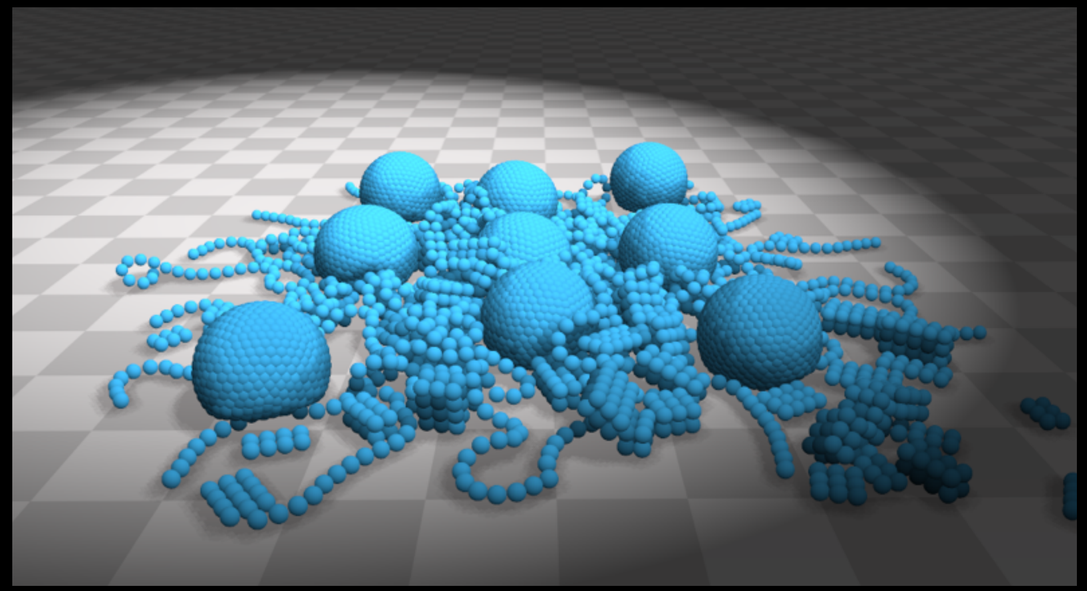


Particles

```
struct Particle
{
  float pos[3];
  float vel[3];
  float invMass;
  int phase;
};
```

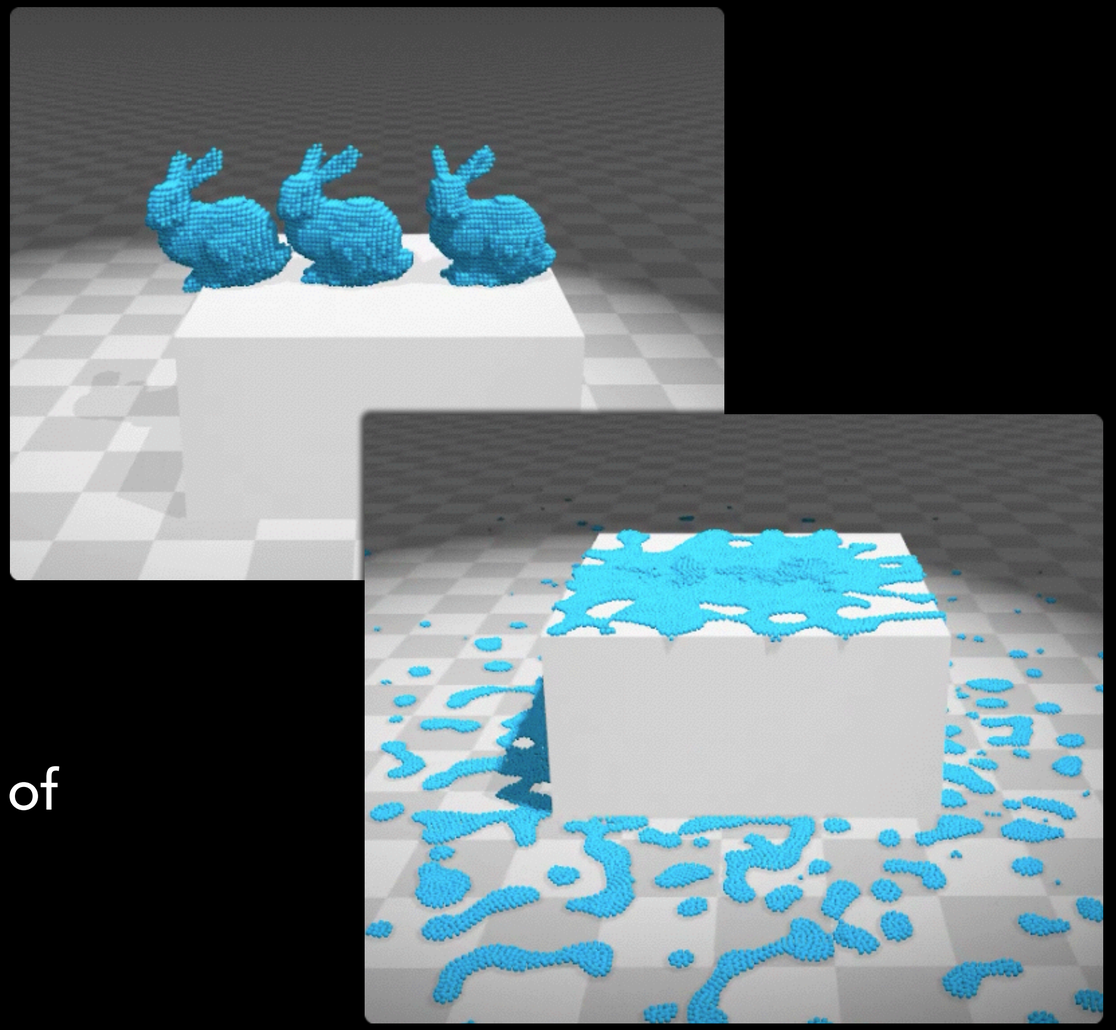
- Phase-ID used to control collision filtering
- Particles do not belong a particular object
- Single collision radius





Constraints

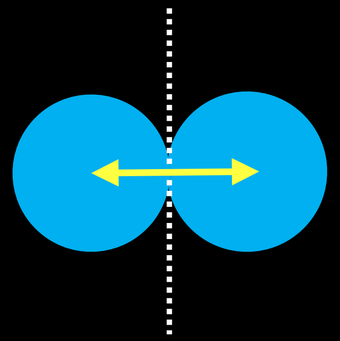
- Constraint types:
  - Distance (clothing)
  - Shape (rigids, plastics)
  - Density (fluids)
  - Volume (inflatables)
  - Contact (non-penetration, friction)
- Combine constraints to create wide variety of effects
  - Melting, phase-changes
  - Stiff cloth, bent metal



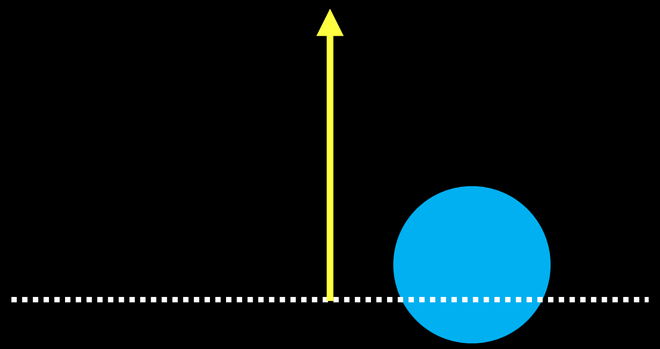
Contact and Friction

Collision Detection

- All dynamics represented as particles
- Kinematic objects represented as meshes
- Two types of collision detection:
  - Particle-Particle
  - Particle-Mesh



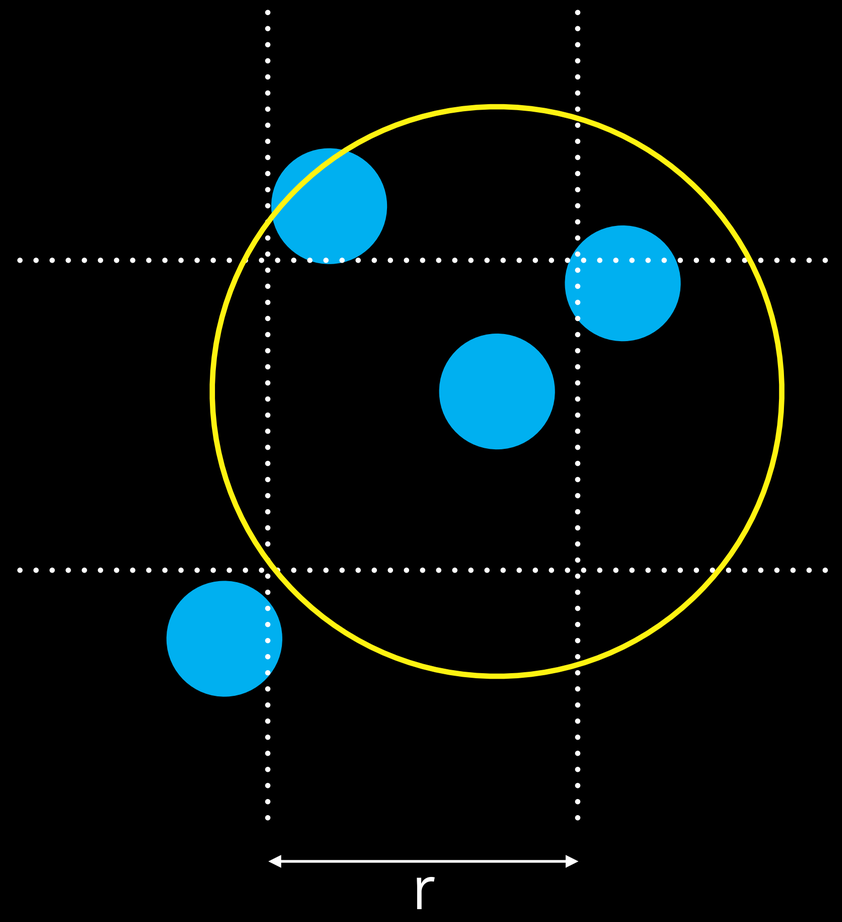
$$C_{contact} = |\mathbf{x}_i - \mathbf{x}_j| - 2r \geq 0$$



$$C_{contact} = \mathbf{n} \cdot \mathbf{x} - r \geq 0$$

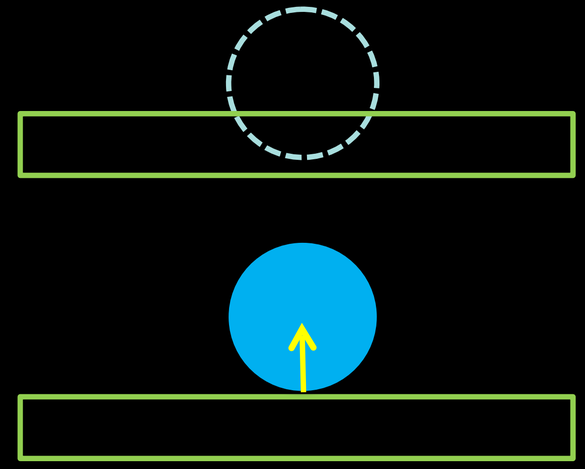
Collision Detection

- Particle-Particle
  - Tiled uniform grid
  - Fixed maximum radius
  - Built using cub::DeviceRadixSort
  - Re-order particle data according to cell index to improve memory locality
  - CUDA Particles Sample [Green 07]



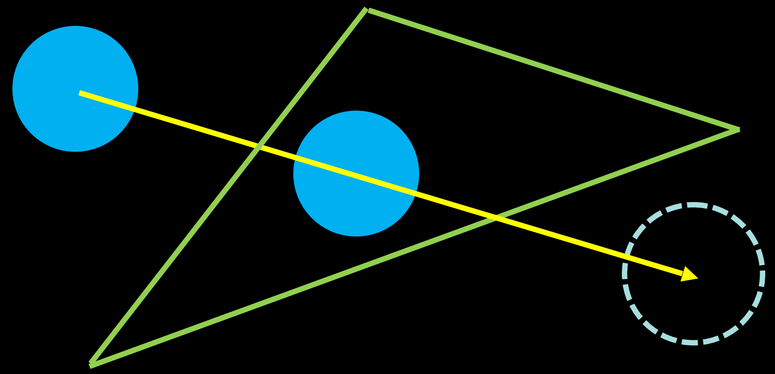
Collision Detection

- Particle-Convex
  - 2D hash-grid
  - Built on GPU
  - 1 warp-per shape, rasterizes projected bounds to grid (~1500 shapes / ms)



Convex Collision (MTD)

- Particle-Triangle Mesh
  - 3D hash-grid
  - Rasterized in CUDA
  - Lollipop test (CCD)



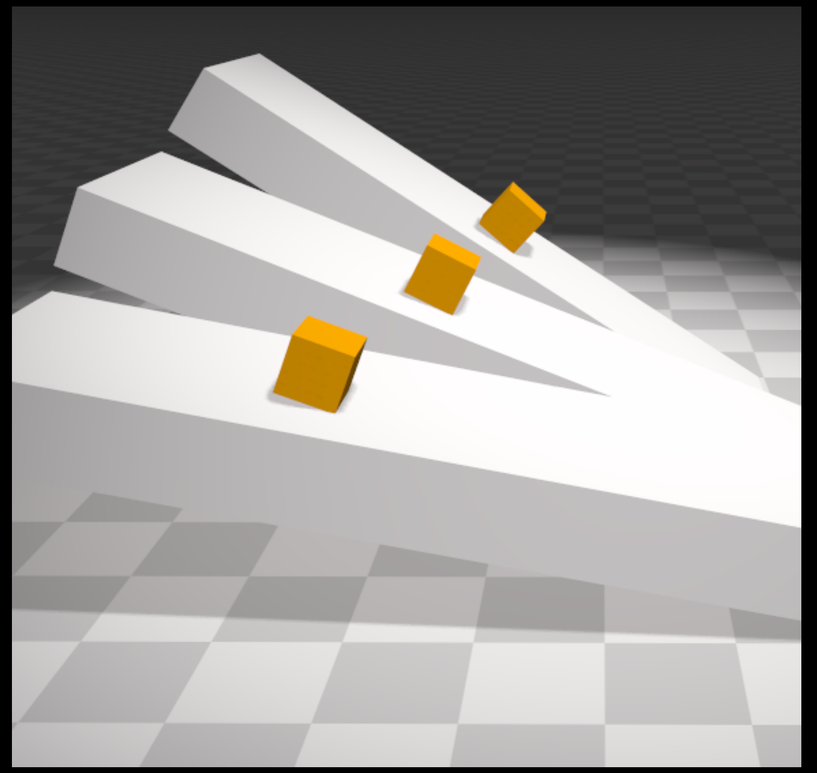
Triangle Collision (TOI)

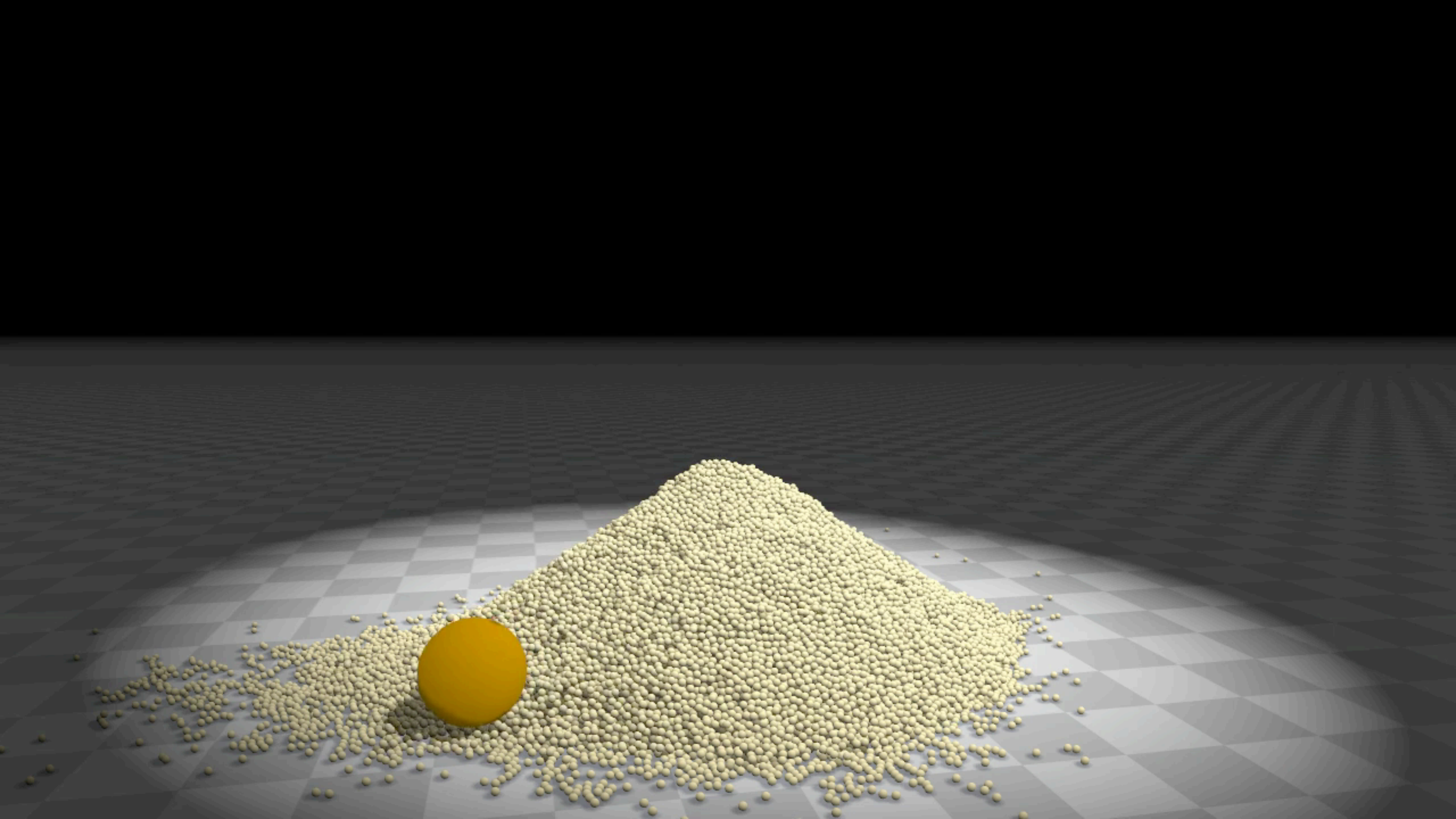
Friction

- Friction in PBD traditionally applied using a velocity filter
- Replace with a position-level frictional constraint

$$C_{friction} = k\,|(\mathbf{x} - \mathbf{x}_0)_{\perp}\mathbf{n}|$$

- Approximate Coulomb friction using penetration depth to limit constraint lambda
- Generates convincing particle piling

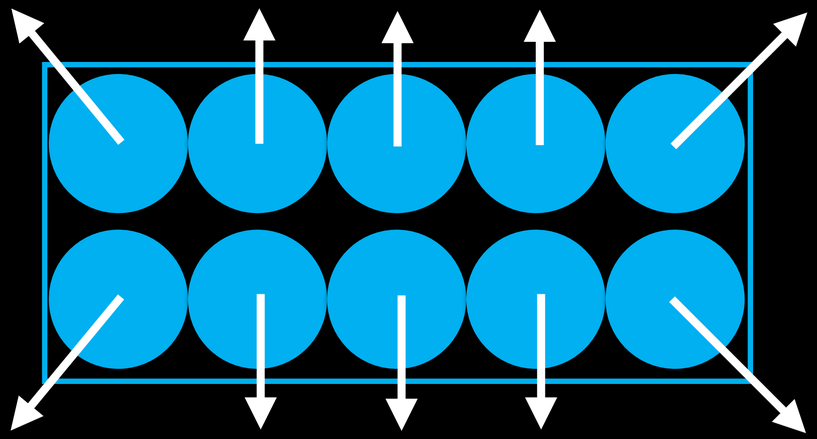


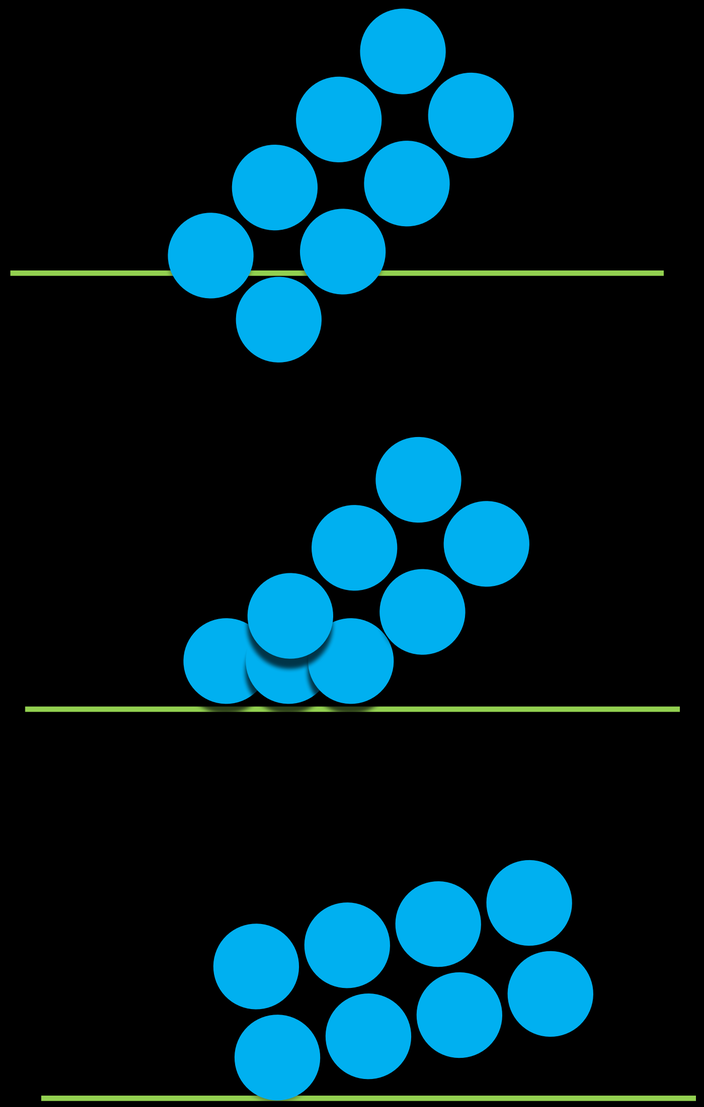


Rigid Bodies

Rigid Bodies

- Convert mesh->SDF
- Place particles in interior
- Add shape-matching constraint
- Store SDF dist + gradient on particles





Rest Configuration

Deformed State

Best Rigid Rotation/ Translation

Shape matching on the GPU

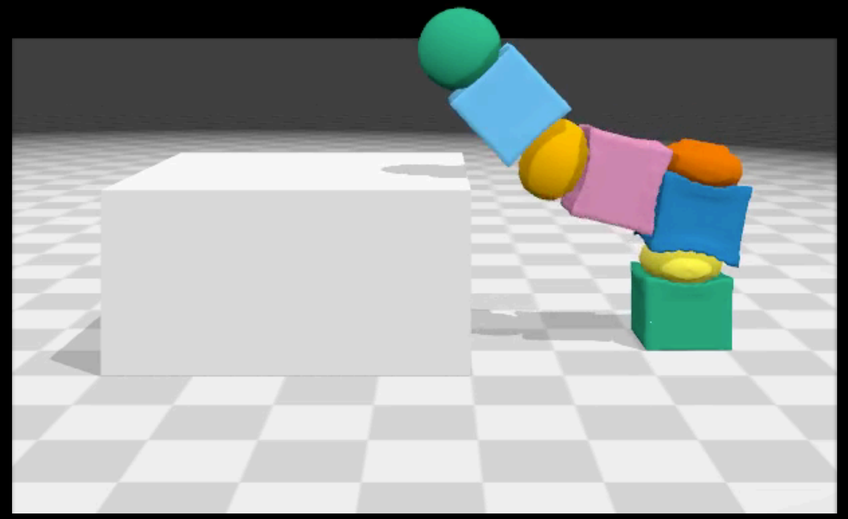
- Shape matching requires computing centre of mass and the moment matrix for particles:

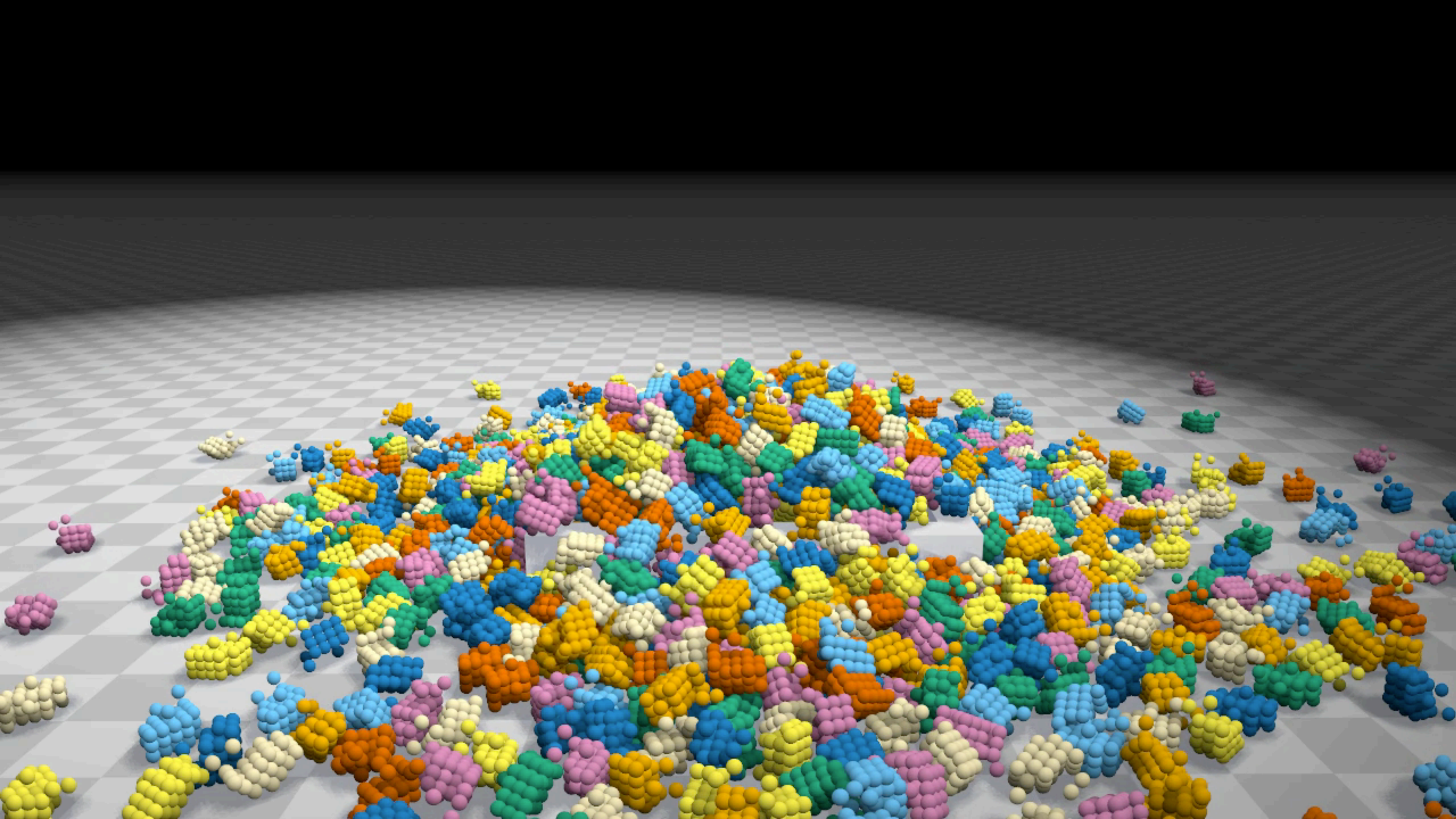
$$\mathbf{c} = \sum_i m_i \mathbf{x_i} / \sum_i m_i \qquad \mathbf{A} = \sum_i m_i (\mathbf{x_i} - \mathbf{c})(\bar{\mathbf{x}}_\mathbf{i} - \bar{\mathbf{c}})^\mathbf{T}$$

- Large summations, not immediately parallel friendly
- Optimized using two parallel cub::BlockReduce calls
- O(N) -> O(logN) (18ms -> 0.6ms)
- 1 block per-rigid shape (64 threads, heuristic, irregular workload problem)
- Polar decomposition still single threaded

Plastic Deformation

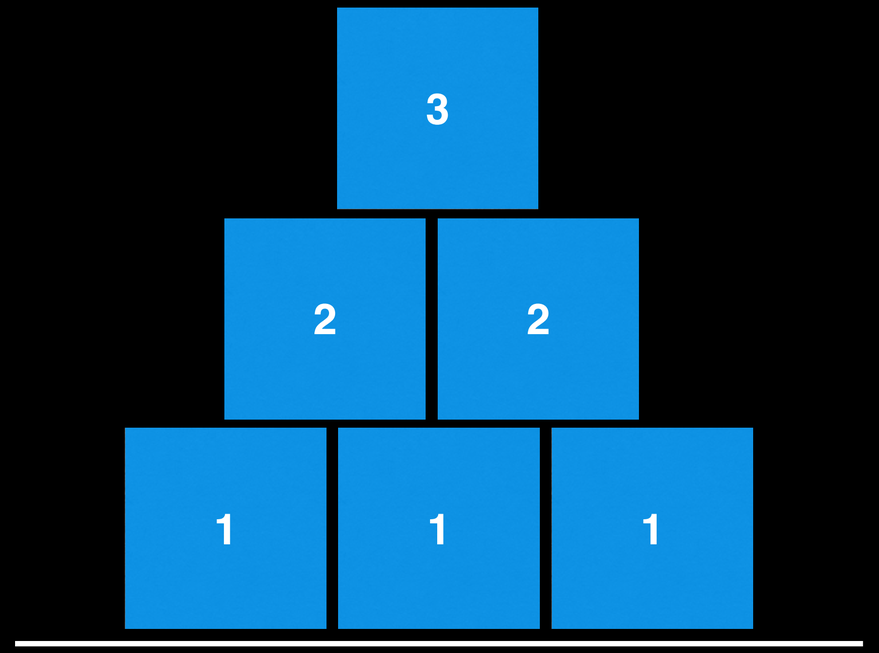
- Detect when deformation exceeds a threshold
- Simply change rest-configuration of particles
- Adjust visual mesh (linear skinning)





Rigid Bodies - Piles and Stacks

- Piles of objects can take many iterations to appear stiff
- Common solution: shock propagation [Guendelman et al. 03]
  - Re-orders constraint solve bottom-top
  - Sets mass of each layer = ∞
  - Problem: limited parallelism

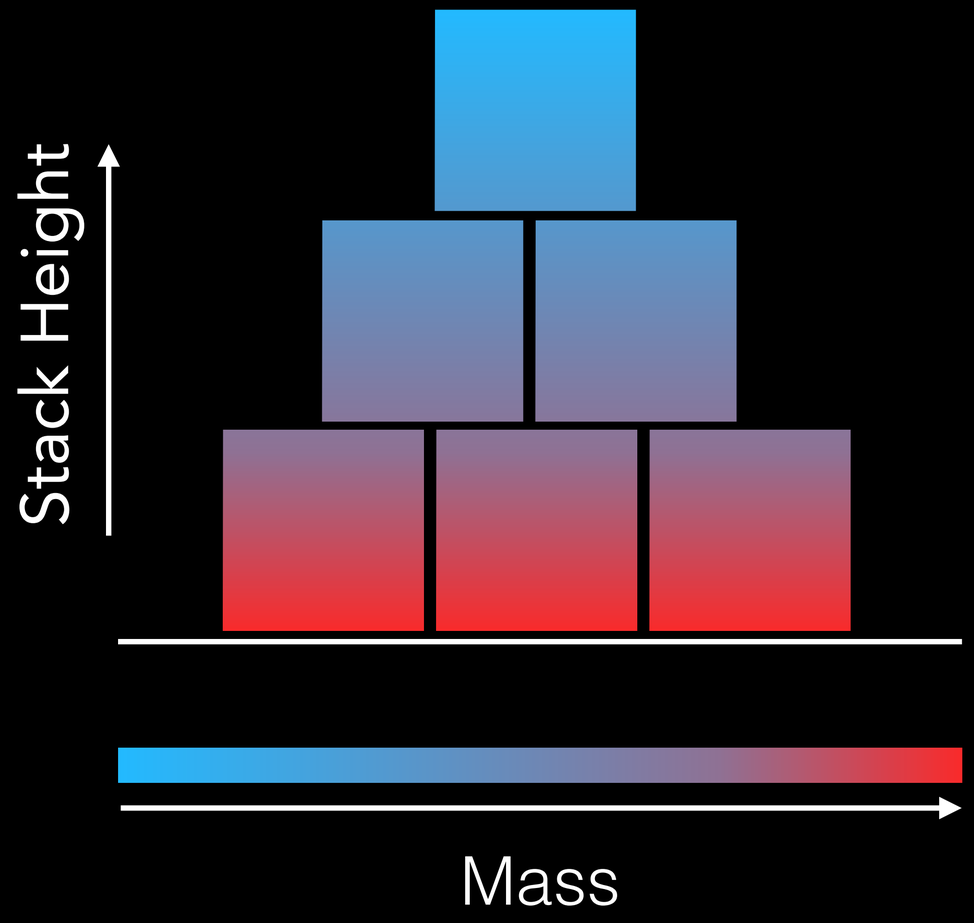


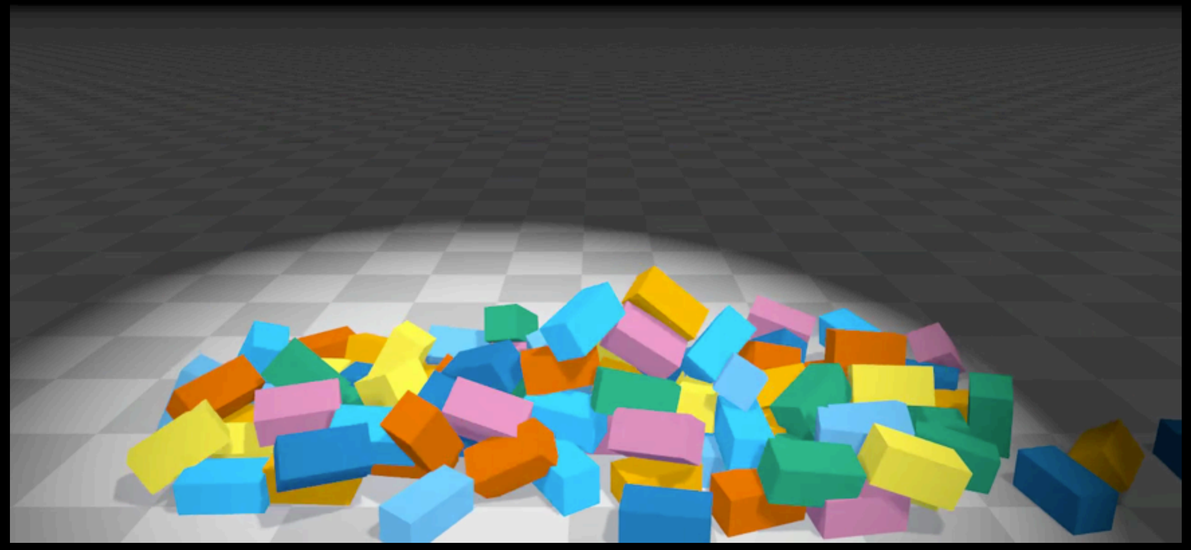
Approximate Shock Propagation

- A parallel friendly solution
- Instead of discrete layers with infinite mass, we modify mass continuously
- Choose ‘stack height’ function and evaluate for each particle:

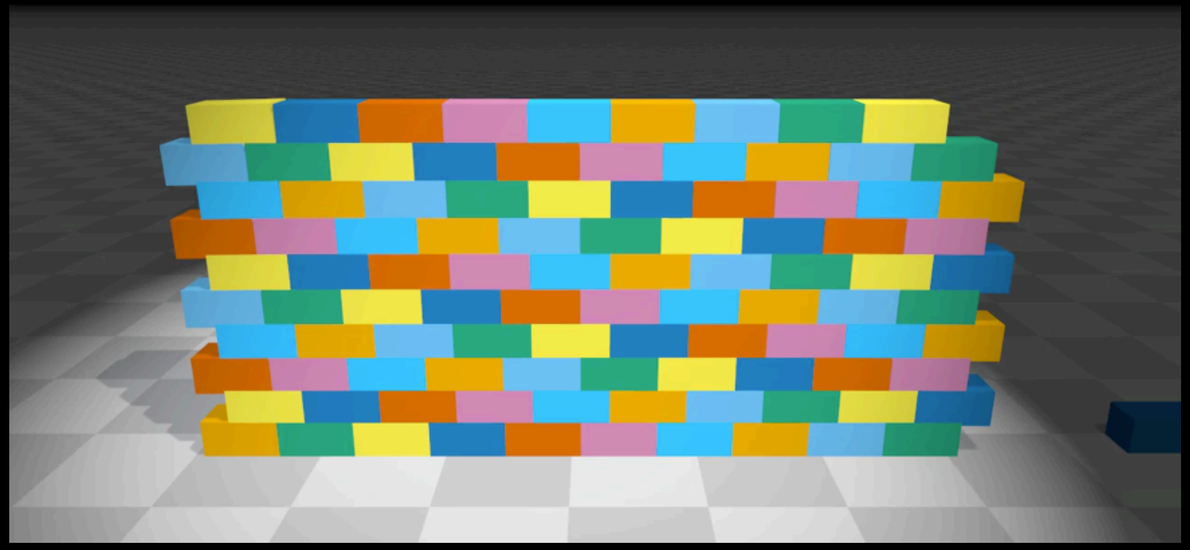
$$height(\mathbf{x}) = |\mathbf{x} - \mathbf{x}_{boundary}|$$

- Temporarily scale particle mass inversely with height





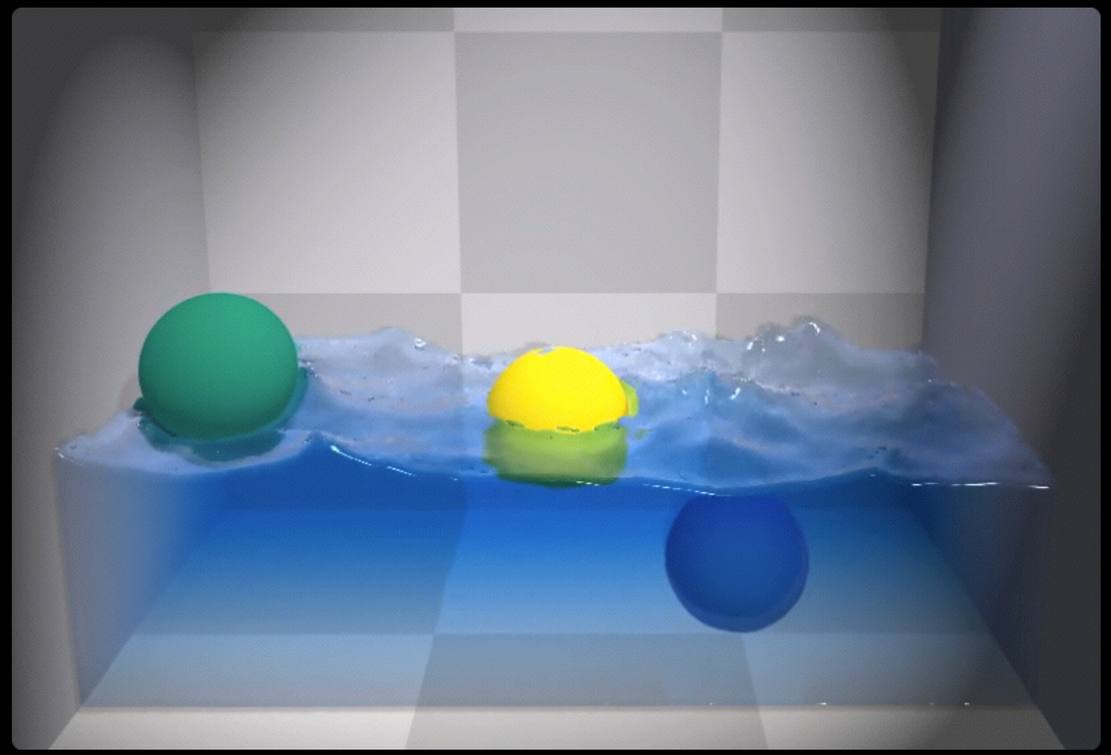
2 Iterations
No Shock Propagation



2 Iterations
With Shock Propagation

Two-Way Rigid Fluid Coupling

- Mostly automatic
- Include all particles in fluid density estimation
- Treat fluid->solid particle interactions as if both particles solid

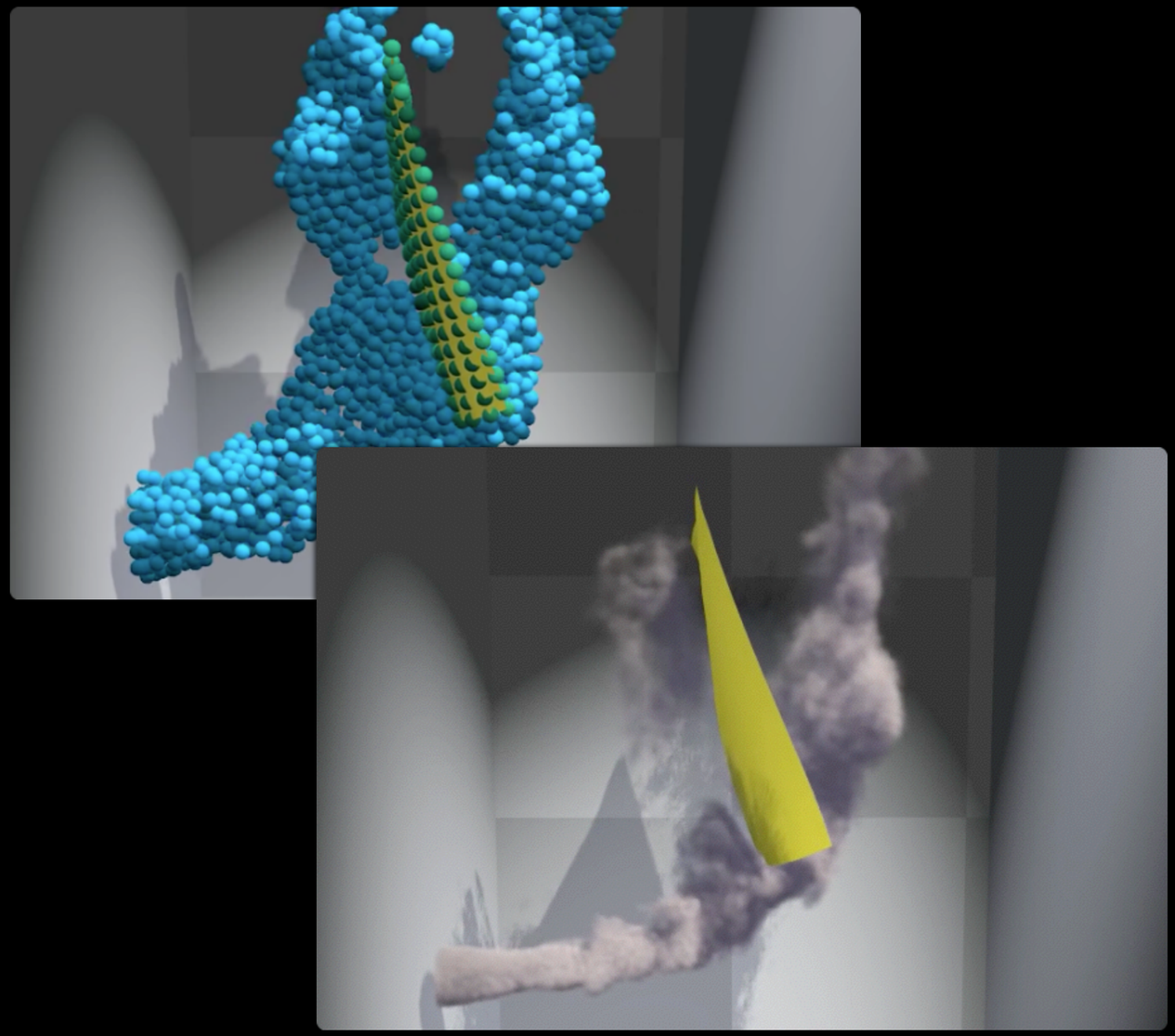




Gases

Gases

- Treat as an incompressible fluid with density constraint
- Sparse representation
- Passive smoke advection (“diffuse particles”)



Gas Forces

Free surface drag

$\bar{\mathbf{f}}_{\mathbf{drag}}$

Density gradient via SPH derivatives:

$$\nabla \rho = \bar{\mathbf{n}}$$

Pressure gradient via Boussinesq approximation:

$$\nabla p = \bar{\mathbf{g}}$$

$\bar{\mathbf{v}}_{\mathbf{0}}$

Baroclinic vorticity:

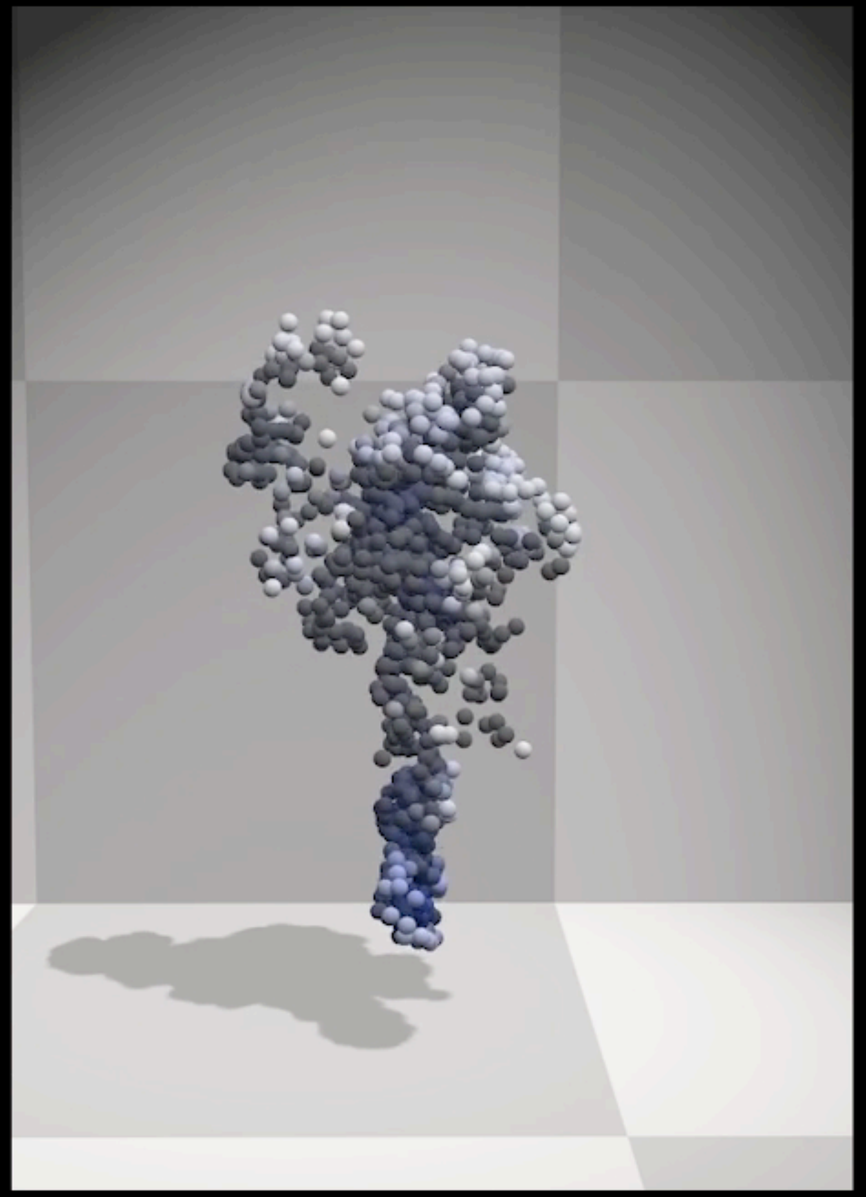
$$\frac{\mathrm{D}\bar{\omega}}{\mathrm{d}t} = \nabla \rho \times \nabla p$$

Driving vorticity:

$$\bar{\mathbf{f}}_{\mathbf{vort}} = \bar{\omega} \times \bar{\mathbf{x}}_{\mathbf{ij}}$$



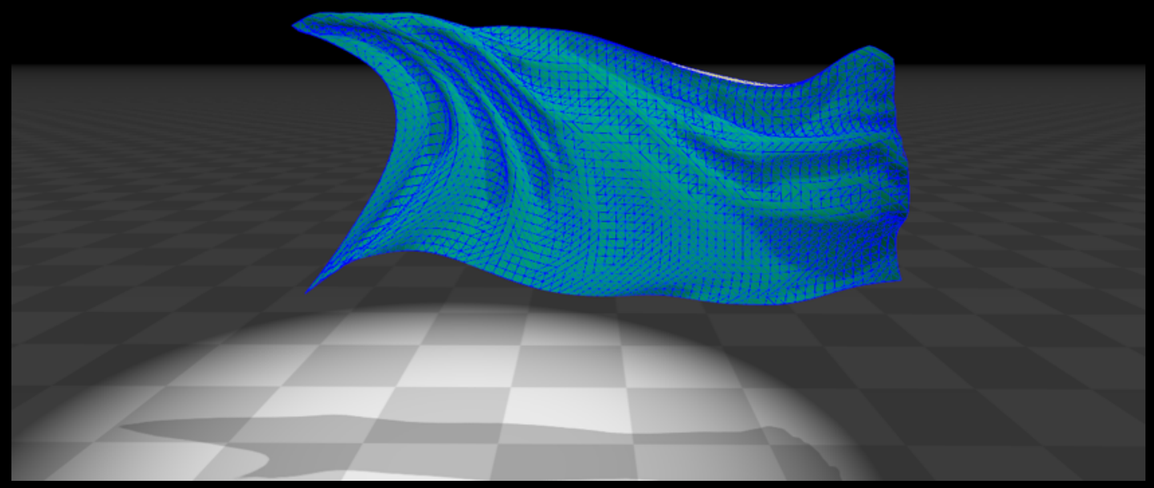
Smoke Particles

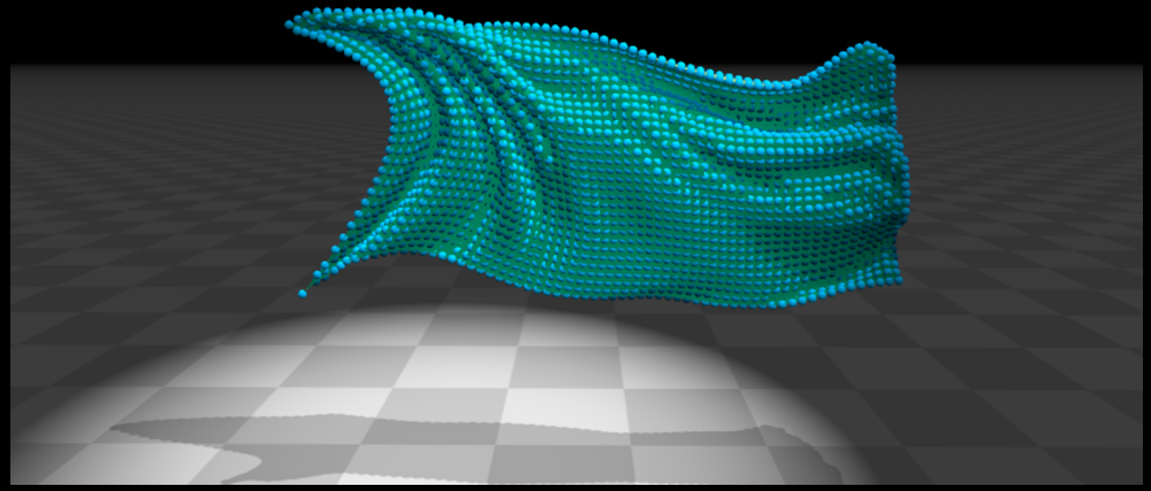


Fluid Particles

Cloth

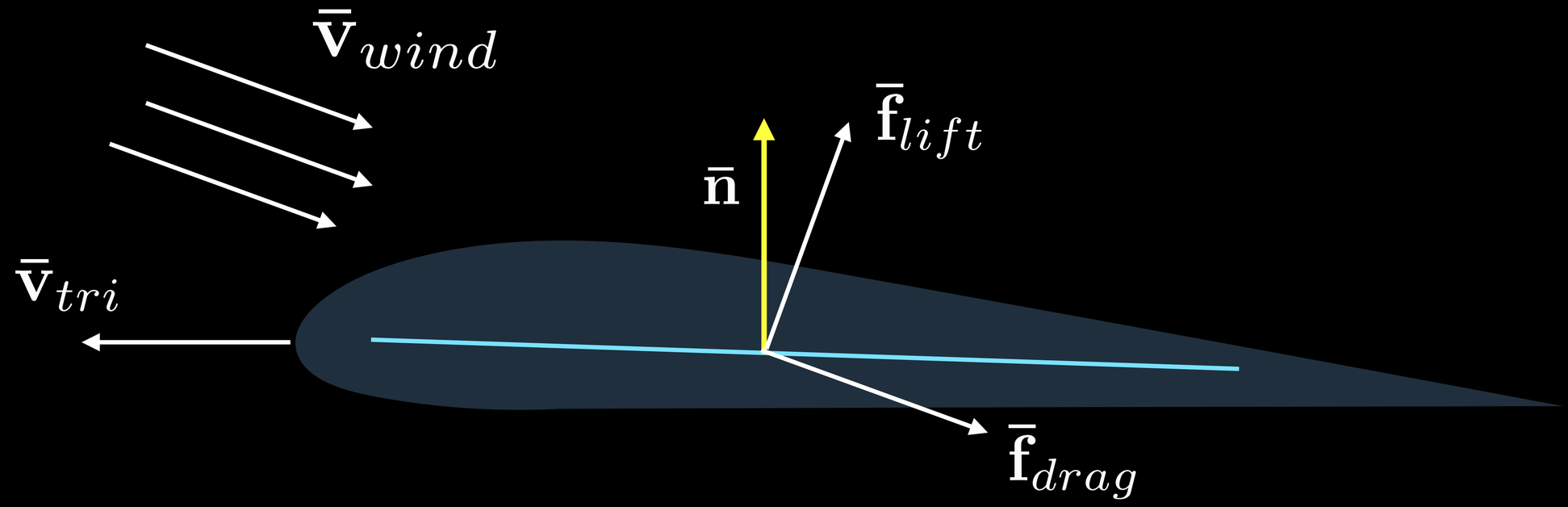
- Graph of distance + tether constraints
- Adding/removing constraints is easy (tearing)
- Self-collision / inter-collision automatically handled

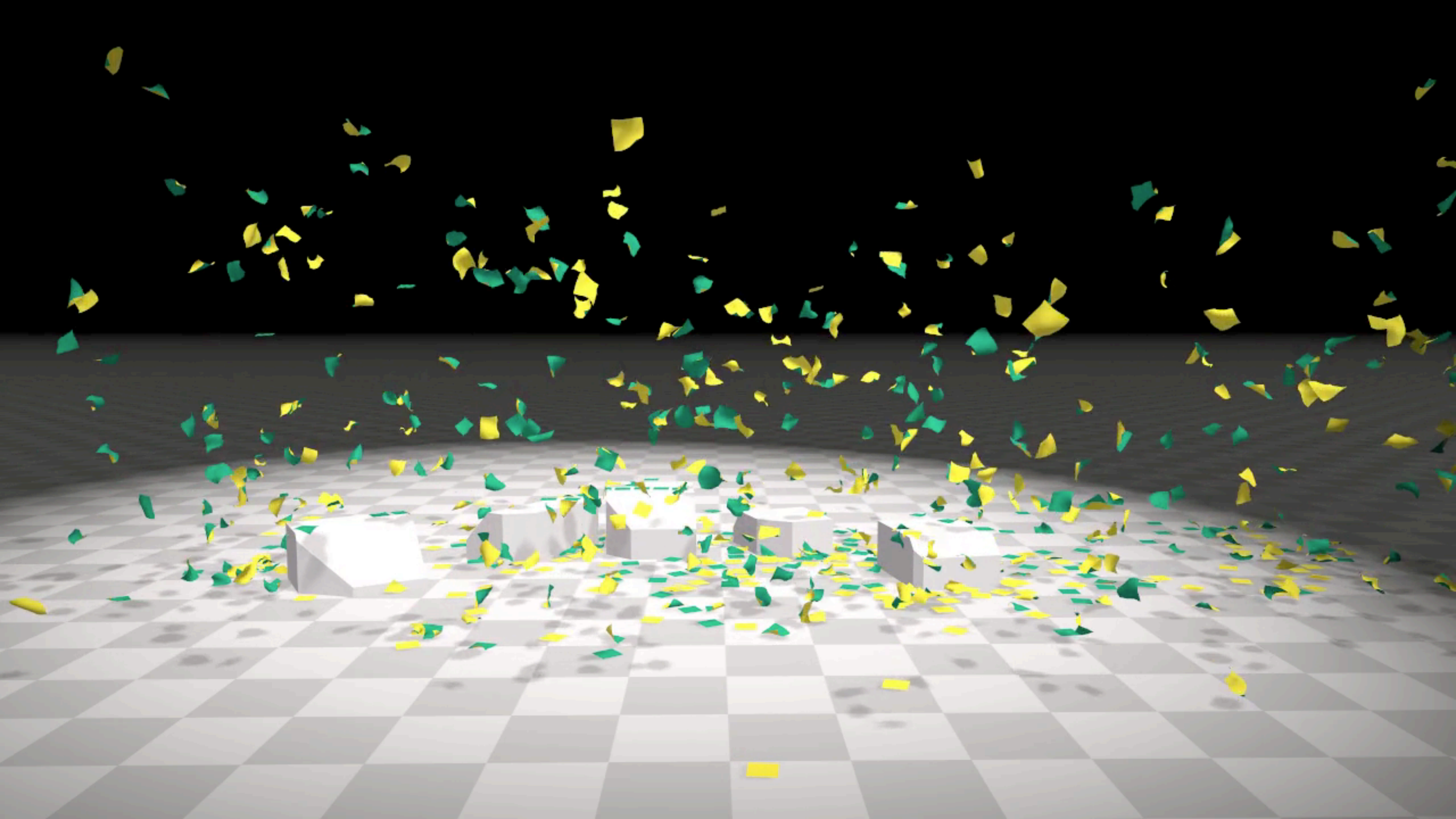




Cloth - Forces

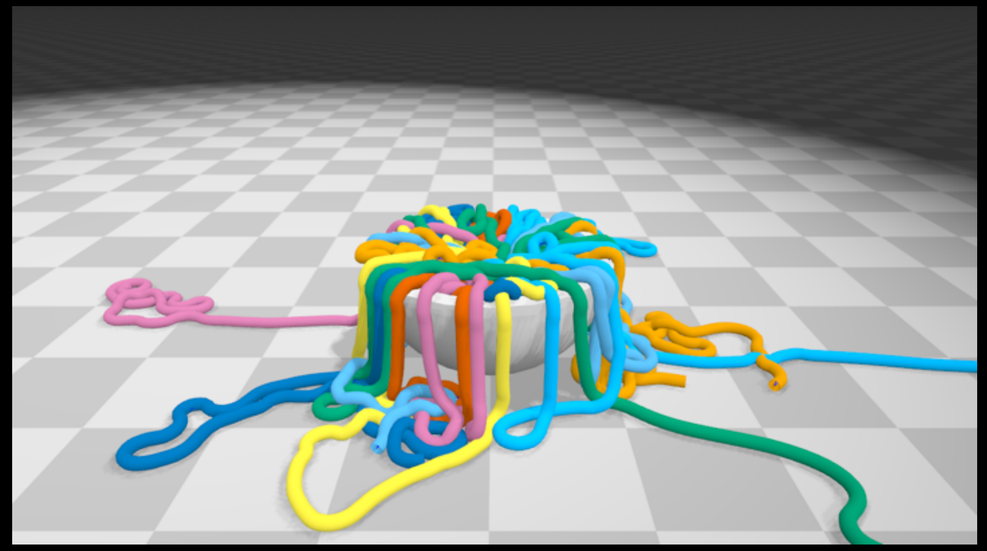
- Basic aerodynamic model
- Treat each triangle as a thin airfoil to generate lift + drag
- Flexible enough to model paper planes

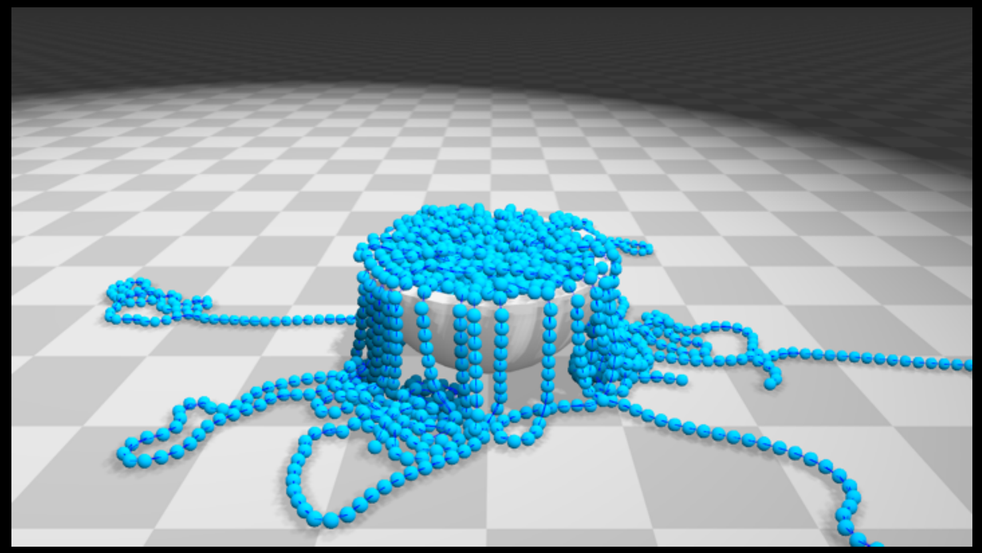


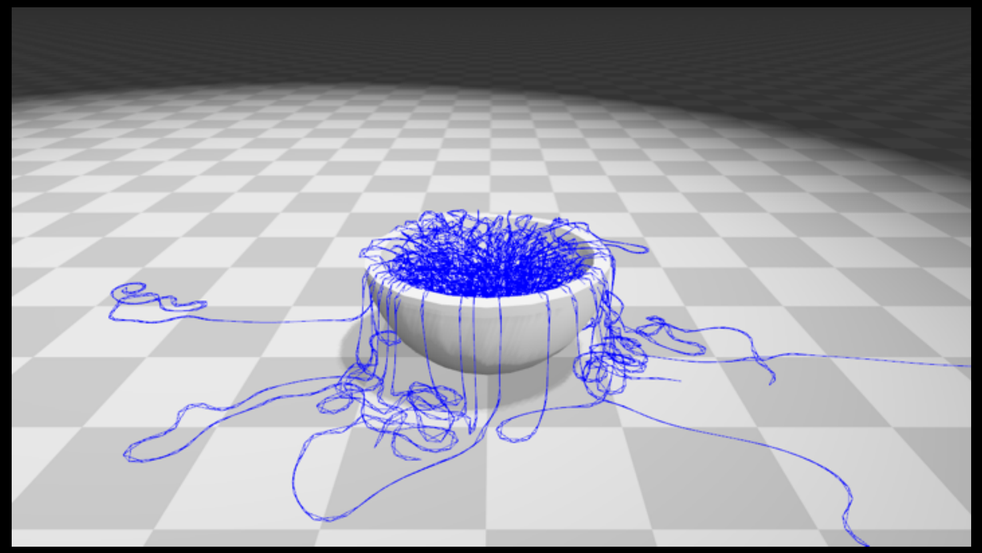


Ropes

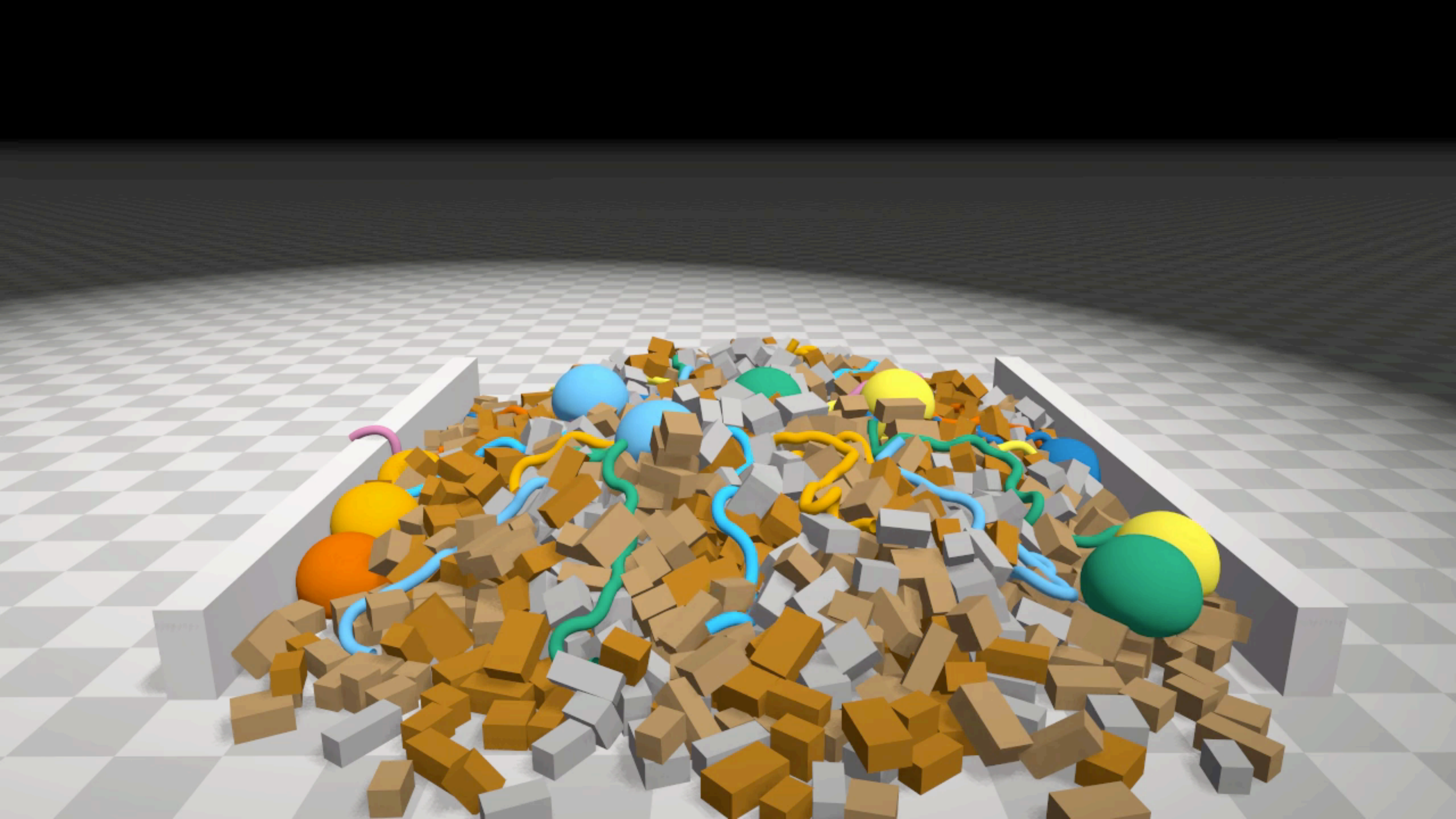
- Build ropes from distance + bending constraints
- Fit Catmull-Rom spline to points
- No torsion constraint (need orientation)

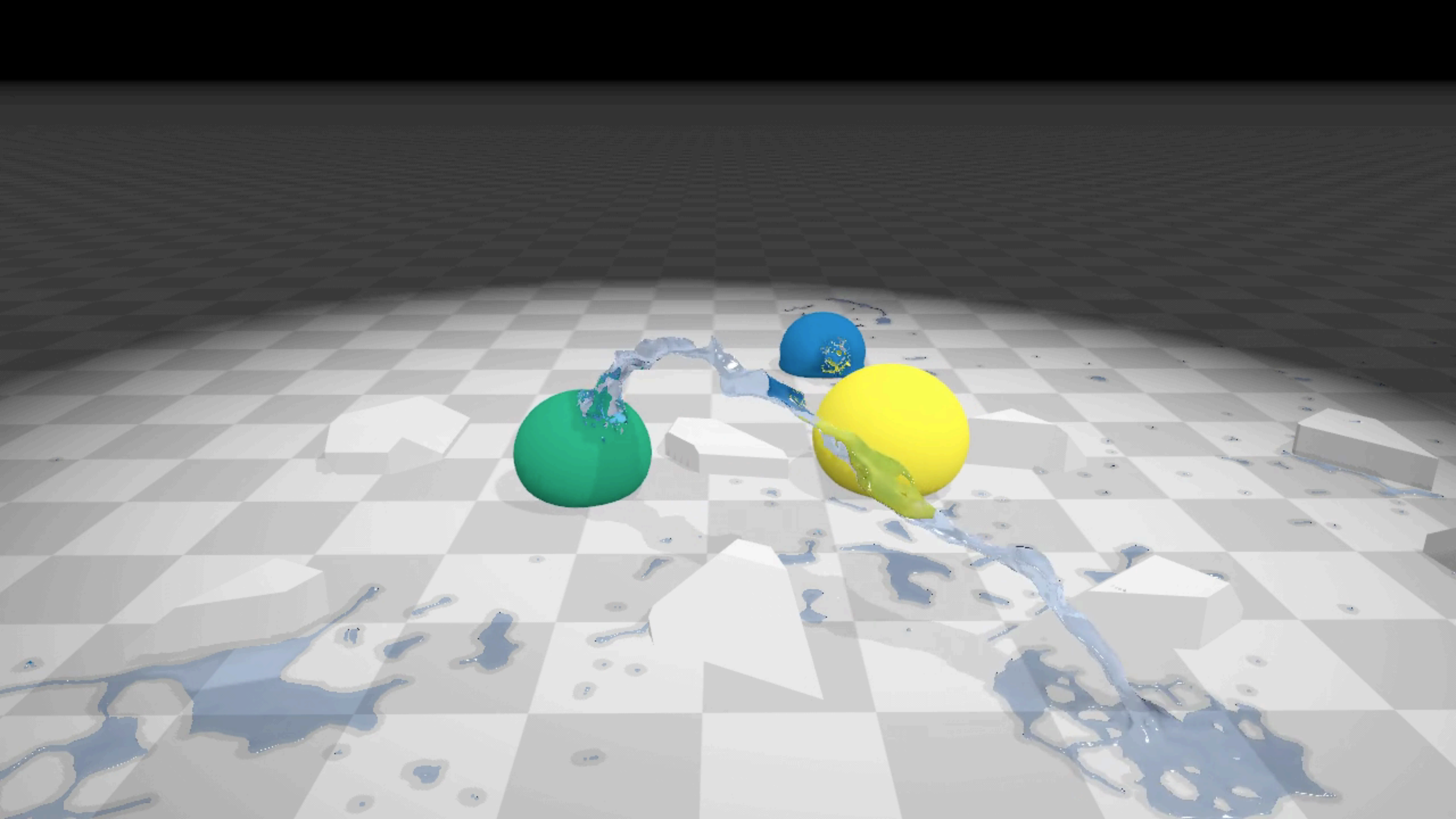






Examples





Limitations / Future Work

- Representing smooth surfaces problematic
- Parallel and robust collision of simplices
- Dynamic sampling for gases
- Convergence for parallel solver
- Shock propagation/LRA are specific hacks, aim to find a more general solution

Questions?

References

- English, Elliot, and Robert Bridson. "Animating developable surfaces using nonconforming elements." ACM Transactions on Graphics (TOG). Vol. 27. No. 3. ACM, 2008.
- Goldenthal, Rony, et al. "Efficient simulation of inextensible cloth." ACM Transactions on Graphics (TOG) 26.3 (2007): 49.
- Bouaziz, Sofien, et al. "Projective dynamics: fusing constraint projections for fast simulation." ACM Transactions on Graphics (TOG) 33.4 (2014): 154.
- Bridson, Robert, Ronald Fedkiw, and John Anderson. "Robust treatment of collisions, contact and friction for cloth animation." ACM Transactions on Graphics (ToG). Vol. 21. No. 3. ACM, 2002.
- Stam, Jos. "Nucleus: Towards a unified dynamics solver for computer graphics." Computer-Aided Design and Computer Graphics, 2009. CAD/Graphics' 09. 11th IEEE International Conference on. IEEE, 2009.
- Green, Simon. "Cuda particles." nVidia Whitepaper 2.3.2 (2008): 1.
- Guendelman, Eran, Robert Bridson, and Ronald Fedkiw. "Nonconvex rigid bodies with stacking." ACM Transactions on Graphics (TOG). Vol. 22. No. 3. ACM, 2003.