Position-Based Dynamics

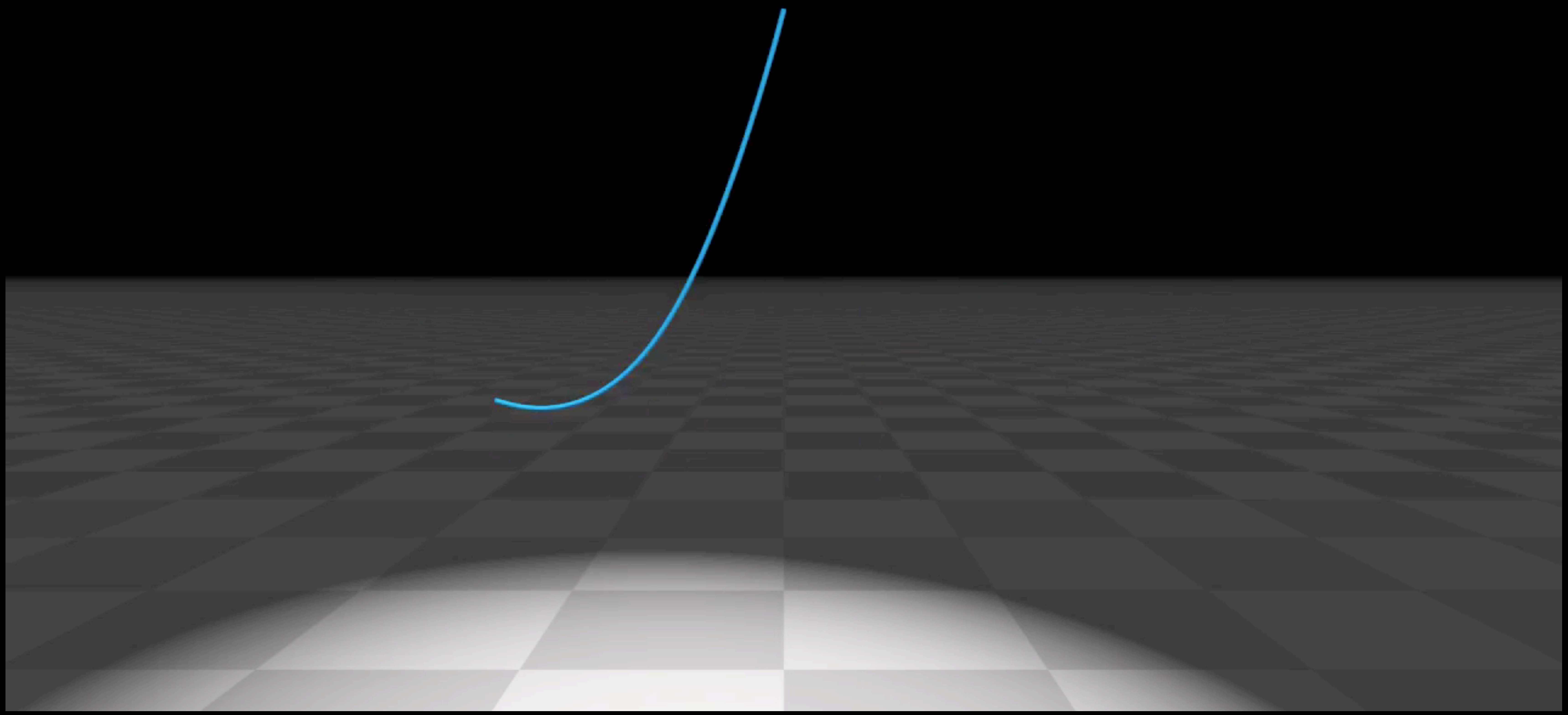
Analysis and Implementation

Miles Macklin

Analysis

Position-Based Dynamics

- Very stable
- Highly damped
- Example



Continuous Equations of Motion

- Newton's second law
- Will consider forces which we can derive from an energy potential E(x)
- Our path: start with implicit Euler and transform it into PBD
- Why implicit Euler? Also highly stable, damped.

$$\mathbf{M}\ddot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$$

Implicit Euler Integration

- Implicit Euler:

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

- Equivalent to:

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

- Forces evaluated at end of the time-step
- Implicit, position-level, time-discretization of Newton's equations

Variational Implicit Euler

- Discrete equations of motion
- Are the first order optimality conditions for a non-linear minimization
- [Goldenthal et al. 2007]
  [Liu et al. 2013]

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

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

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

Variational Implicit Euler

- In the limit of infinite stiffness we obtain a constrained minimization

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

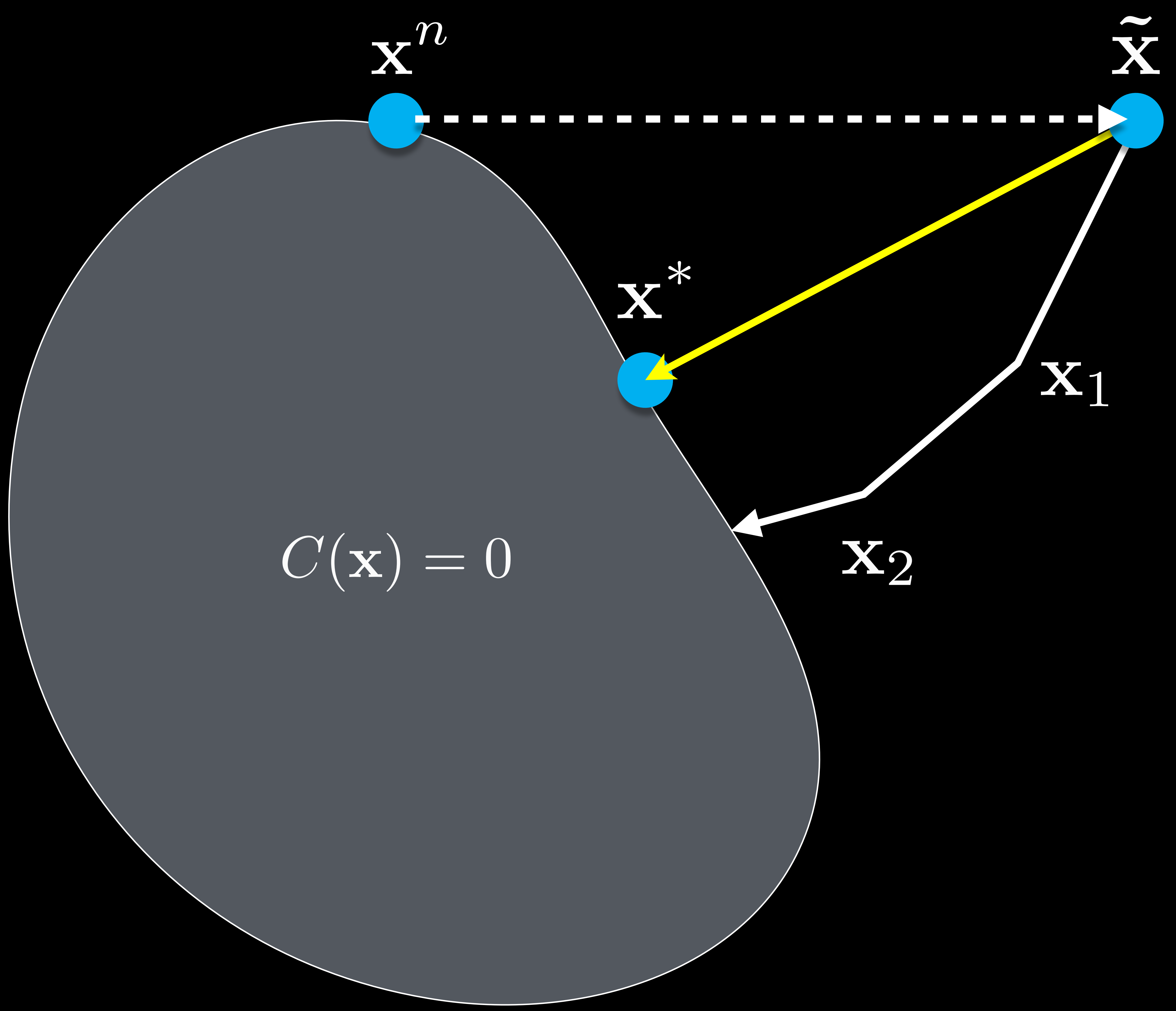
$$E \rightarrow \infty$$

$$\begin{aligned} \operatorname{argmin} \quad & \frac{1}{2}(\mathbf{x}^{n+1} - \tilde{\mathbf{x}})^T \mathbf{M} (\mathbf{x}^{n+1} - \tilde{\mathbf{x}}) \\ \text{subject to} \quad & \mathbf{C}(\mathbf{x}^{n+1}) = 0 \end{aligned}$$

Geometric Interpretation

$$\begin{array}{ll} \text{argmin} & \frac{1}{2}(\mathbf{x}^{n+1} - \tilde{\mathbf{x}})^T \mathbf{M}(\mathbf{x}^{n+1} - \tilde{\mathbf{x}}) \\ \text{subject to} & \mathbf{C}(\mathbf{x}^{n+1}) = 0 \end{array}$$

- Variational form gives a “step and project” interpretation for implicit Euler
- PBD performs approximate projection



Solving

- Implicit time discretization produces a non-linear system of equations
- How do we solve such a system?
- Newton's method

Discrete constrained equations of motion

$$\mathbf{M}(\mathbf{x}^{n+1} - \tilde{\mathbf{x}}) - \Delta t^2 \nabla \mathbf{C}(\mathbf{x}^{n+1})^T \boldsymbol{\lambda} = \mathbf{0}$$
$$\mathbf{C}(\mathbf{x}^{n+1}) = \mathbf{0}$$

Non-Linear System

$$\mathbf{g}(\mathbf{x}_i, \boldsymbol{\lambda}_i) = \mathbf{0}$$
$$\mathbf{h}(\mathbf{x}_i, \boldsymbol{\lambda}_i) = \mathbf{0}$$

Approximate Newton Step

First approximation:

- M = K + O(dt^2)
- Common Quasi-Newton simplification

Second approximation:

- Assume g = 0
- True for first iteration
- Typically remains small

Full Newton System

$$\begin{bmatrix} \mathbf{K} & \nabla\mathbf{C}^T \\ \nabla\mathbf{C} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{x} \\ \Delta\boldsymbol{\lambda} \end{bmatrix} = - \begin{bmatrix} \mathbf{g}(\mathbf{x}_i, \boldsymbol{\lambda}_i) \\ \mathbf{h}(\mathbf{x}_i, \boldsymbol{\lambda}_i) \end{bmatrix}$$

Approximate System

$$\begin{bmatrix} \mathbf{M} & \nabla\mathbf{C}^T \\ \nabla\mathbf{C} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{x} \\ \Delta\boldsymbol{\lambda} \end{bmatrix} = - \begin{bmatrix} \mathbf{0} \\ \mathbf{h}(\mathbf{x}_i, \boldsymbol{\lambda}_i) \end{bmatrix}$$

(Schur Complement)

PBD System

$$\left[\nabla\mathbf{C}(\mathbf{x}_i)\mathbf{M}^{-1}\nabla\mathbf{C}(\mathbf{x}_i)^T\right]\Delta\boldsymbol{\lambda} = -\mathbf{C}(\mathbf{x}_i)$$

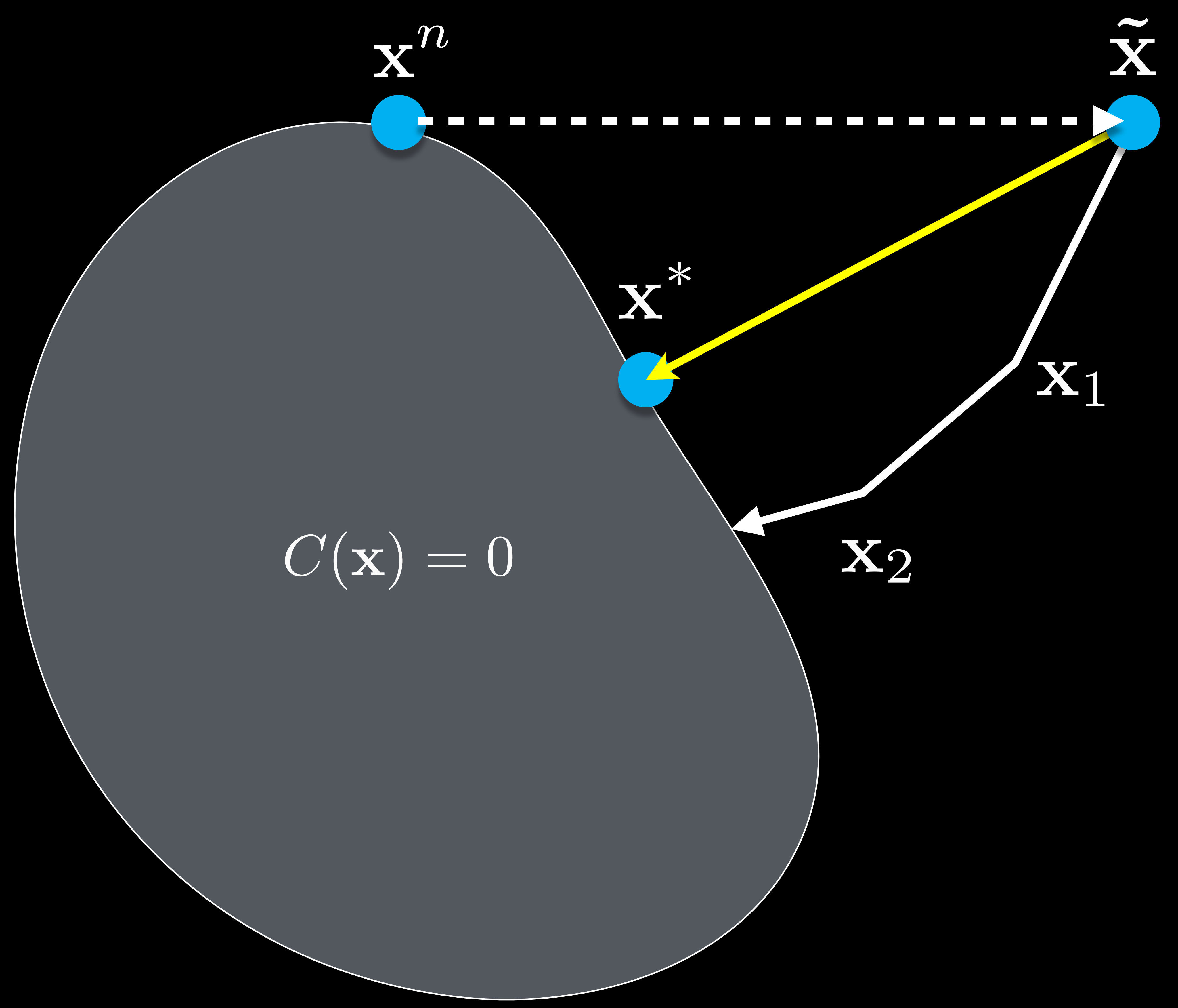
Variational Interpretation of Approximate Projection

Implicit Euler

$$\begin{aligned} \text{argmin} \quad & \frac{1}{2}(\mathbf{x} - \tilde{\mathbf{x}})^T \mathbf{M}(\mathbf{x} - \tilde{\mathbf{x}}) \\ \text{subject to} \quad & \mathbf{C}(\mathbf{x}) = \mathbf{0} \end{aligned}$$

PBD (each iteration)

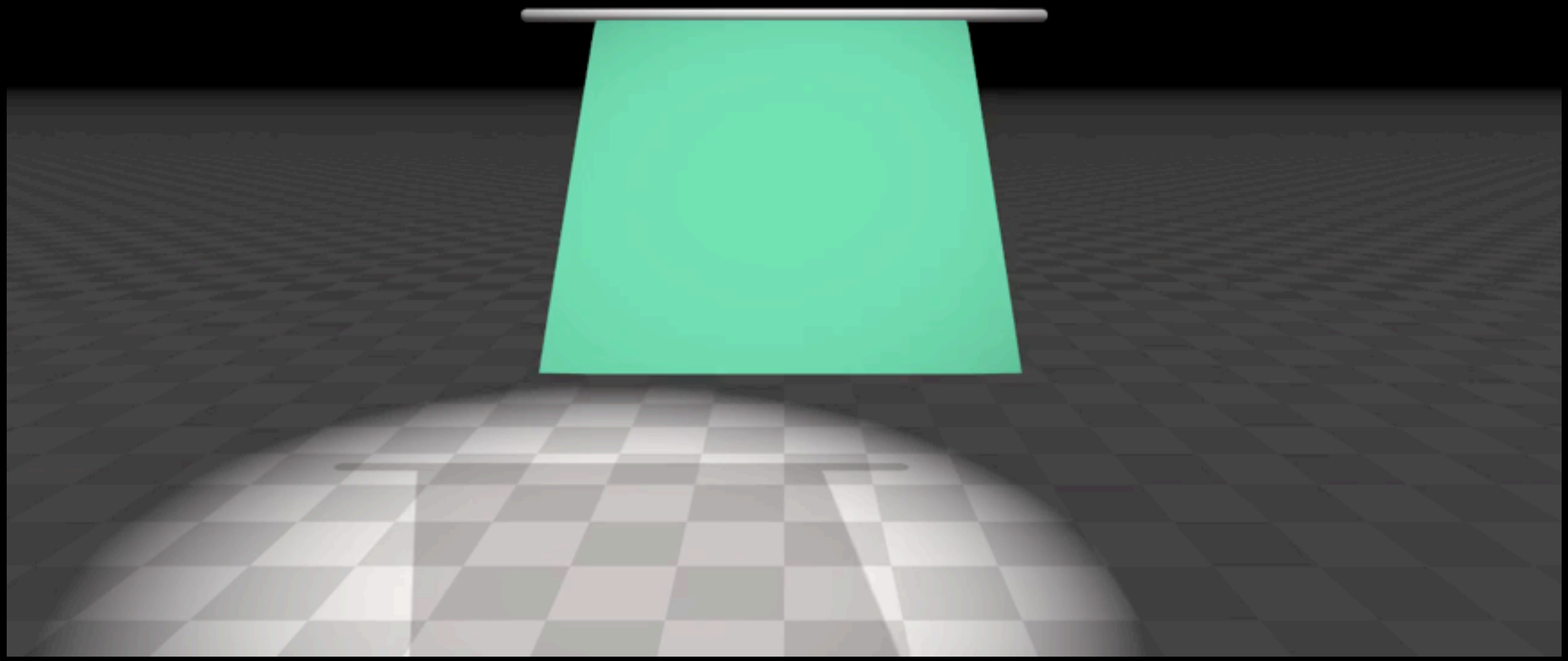
$$\begin{aligned} \text{argmin} \quad & \frac{1}{2}(\mathbf{x} - \mathbf{x}_i)^T \mathbf{M}(\mathbf{x} - \mathbf{x}_i) \\ \text{subject to} \quad & \mathbf{C}(\mathbf{x}) = \mathbf{0} \end{aligned}$$



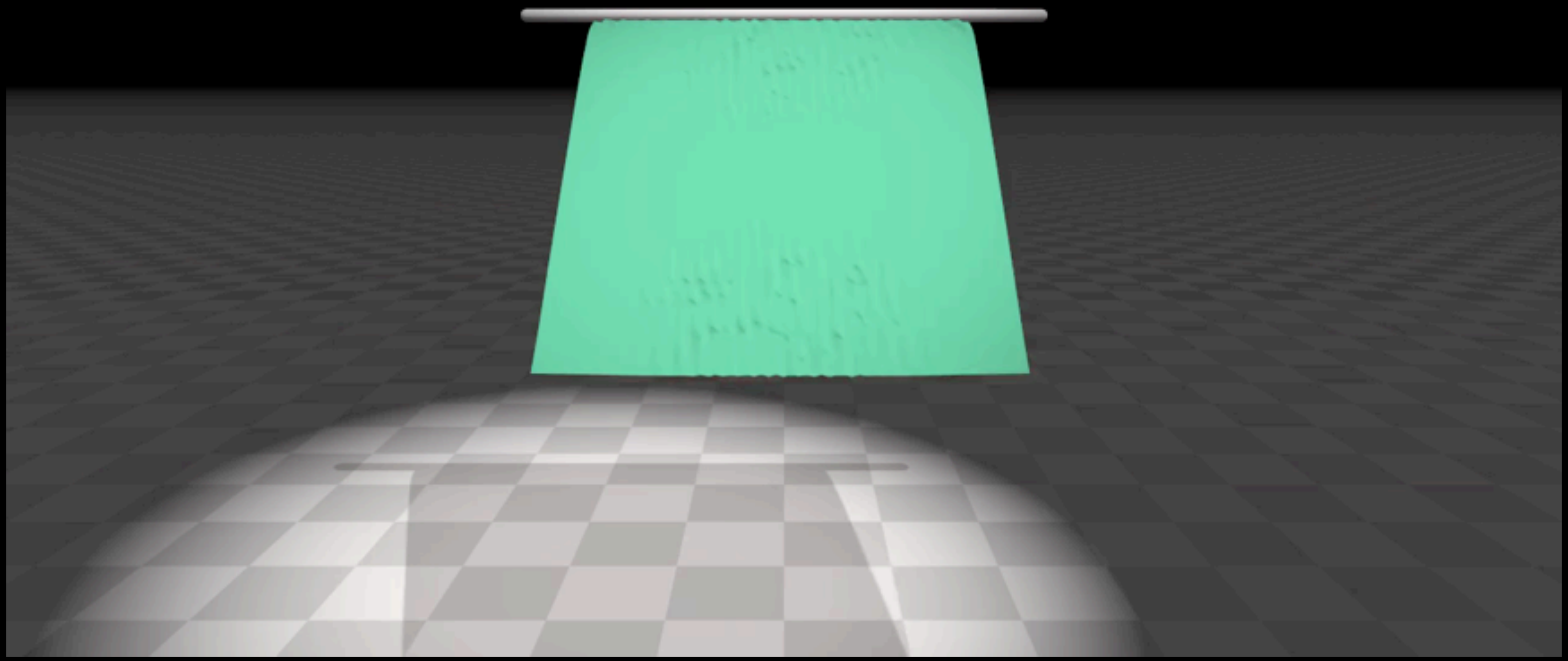
Problems

- To arrive at PBD we had to assume infinitely stiff energy potentials
- This means PBD converges to an infinitely stiff solution regardless of stiffness coefficient
- Stiffness dependent on iteration count and time-step
- No concept of total constraint force
- Fully implicit -> severe energy dissipation

Iteration Count Dependent Stiffness



20 ITERATIONS



160 ITERATIONS

PBD Extensions

- Projective Dynamics [Bouaziz et al. 2014]
- XPBD [Macklin et al. 2016]
- Second order PBD

XPBD

- Instead of assuming infinite stiffness, allow constraints to be compliant
- Leads to a modified / regularized non-linear system
- Direct correspondence to engineering stiffness (Young's modulus)
- Compliance is simply inverse stiffness
- [Servin et al. 2006]

Potential

$$E = \frac{1}{2}\mathbf{C}^T(\mathbf{x}^{n+1})\boldsymbol{\alpha}^{-1}\mathbf{C}(\mathbf{x}^{n+1})$$

Compliance

$$\boldsymbol{\alpha} = \mathbf{k}^{-1}$$

NVIDIA

XPBD Newton Step

- Take Schur complement of approximate system with respect to M
- Obtain PBD or Fast Projection form
- [Goldenthal et al 2007]

Modified Newton System

$$\begin{bmatrix} \mathbf{M} & \nabla \mathbf{C}^T \\ \nabla \mathbf{C} & \tilde{\alpha} \end{bmatrix} \begin{bmatrix} \Delta \mathbf{x} \\ \Delta \boldsymbol{\lambda} \end{bmatrix} = - \begin{bmatrix} \mathbf{0} \\ \mathbf{h}(\mathbf{x}_i, \boldsymbol{\lambda}_i) \end{bmatrix}$$

Schur complement

$$\left[\nabla \mathbf{C}(\mathbf{x}_i) \mathbf{M}^{-1} \nabla \mathbf{C}(\mathbf{x}_i)^T + \tilde{\alpha} \right] \Delta \boldsymbol{\lambda} = -\mathbf{C}(\mathbf{x}_i) - \tilde{\alpha} \boldsymbol{\lambda}_i$$

XPBD Gauss-Seidel Update

- View PBD "scaling fator" *s* as incremental Lagrange multiplier
- Additional compliance terms
- Must store Lagrange multiplier for each constraint
- PBD solves the infinite stiffness case

PBD

$$s_j = \frac{-C_j(\mathbf{x}_i)}{\nabla C_j \mathbf{M}^{-1} \nabla C_j^T}$$

XPBD

$$\Delta\lambda_j = \frac{-C_j(\mathbf{x}_i) - \tilde{\alpha}_j \lambda_{ij}}{\nabla C_j \mathbf{M}^{-1} \nabla C_j^T + \tilde{\alpha}_j}$$

XPBD Algorithm

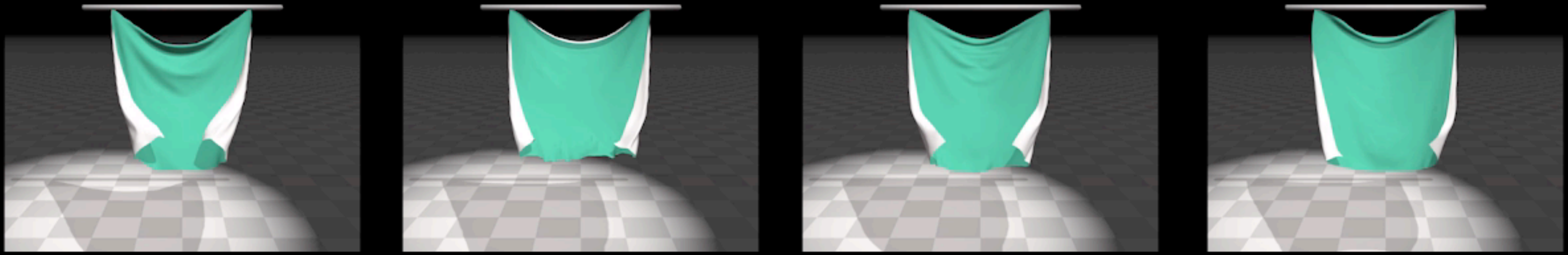
- Only two differences from PBD:
  - Lagrange multiplier calculation (include compliance terms)
  - Lagrange multiplier update (store instead of discard)

predict position $\tilde{\mathbf{x}} \Leftarrow \mathbf{x}^n + \Delta t \mathbf{v}^n + \Delta t^2 \mathbf{M}^{-1} \mathbf{f}_{ext}(\mathbf{x}^n)$

initialize solve $\mathbf{x}_0 \Leftarrow \tilde{\mathbf{x}}$
initialize multipliers $\boldsymbol{\lambda}_0 \Leftarrow \mathbf{0}$
while $i < solverIterations$ **do**
  for all constraints **do**
    compute $\Delta\lambda$
    compute $\Delta\mathbf{x}$
    update $\lambda_{i+1} \Leftarrow \lambda_i + \Delta\lambda$
    update $\mathbf{x}_{i+1} \Leftarrow \mathbf{x}_i + \Delta\mathbf{x}$
  end for
  $i \Leftarrow i + 1$
end while

update positions $\mathbf{x}^{n+1} \Leftarrow \mathbf{x}_i$
update velocities $\mathbf{v}^{n+1} \Leftarrow \frac{1}{\Delta t}\left(\mathbf{x}^{n+1} - \mathbf{x}^n\right)$

Our Method



20 iterations | 40 iterations | 80 iterations | 160 iterations

XPBD - FEM

- Generalizes to arbitrary constitutive models
- Treat strain as vector of constraints
- Compliance matrix is inverse stiffness

Elastic Energy Potential

$$E_{tri} = V \frac{1}{2} \boldsymbol{\epsilon}^T \mathbf{K} \epsilon$$

Constraint Vector

$$\mathbf{C}_{tri}(\mathbf{x}) = \boldsymbol{\epsilon}_{tri} = \begin{bmatrix} \epsilon_x \\ \epsilon_y \\ \epsilon_{xy} \end{bmatrix}$$

Compliance Matrix

$$\boldsymbol{\alpha}_{tri} = \mathbf{K}^{-1} = \begin{bmatrix} \lambda + 2\mu & \lambda & 0 \\ \lambda & \lambda + 2\mu & 0 \\ 0 & 0 & 2\mu \end{bmatrix}^{-1}$$

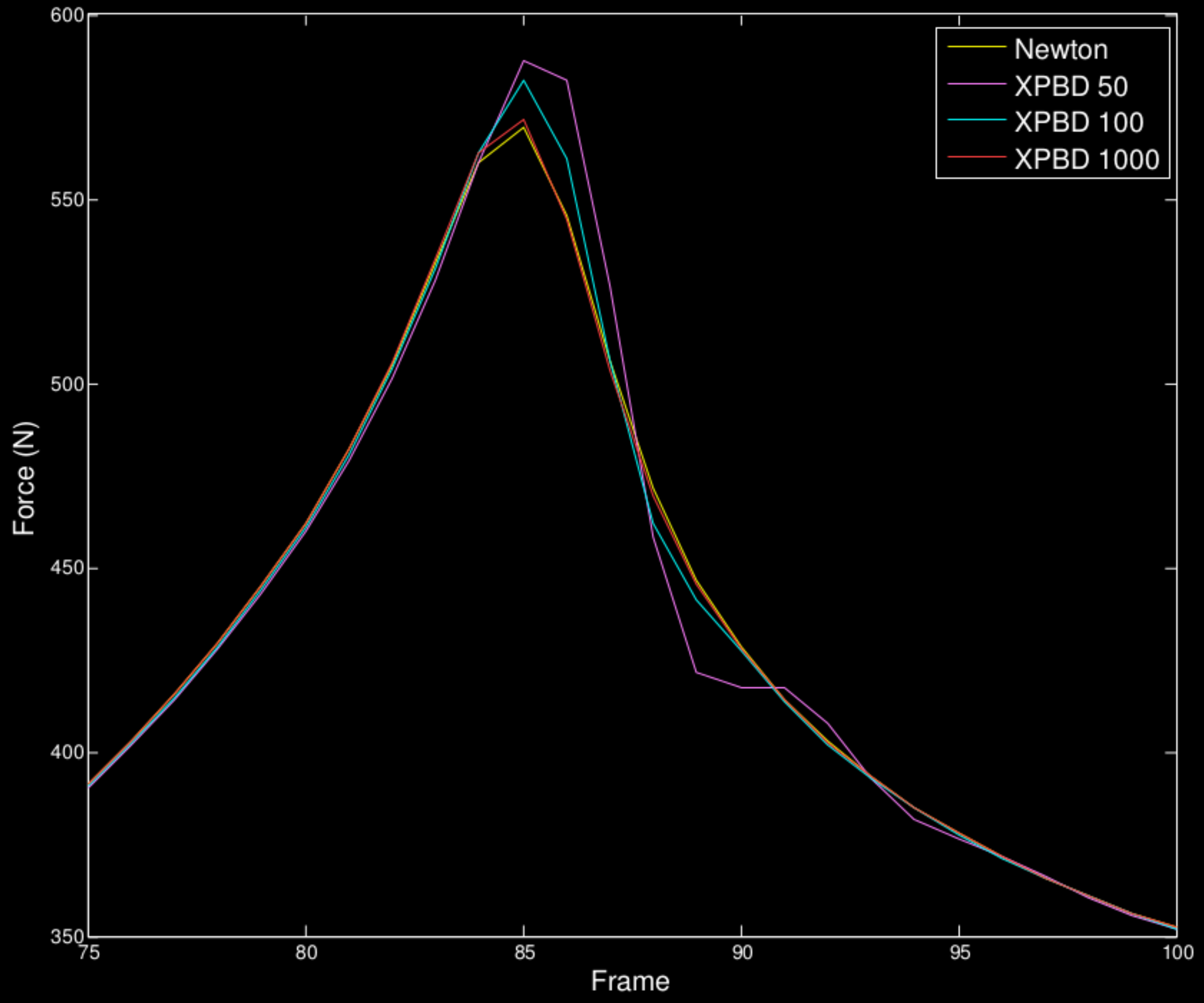
Cantilever Beam

St.Venant-Kirchhoff Triangular FEM

Young's Modulus: E=10^5
Poisson's Ratio: Mu=0.3

Results - XPBD vs Implicit Euler

- Compare solver output to a non-linear Newton method
- Close agreement for primal and dual variables



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:

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

- First order velocity update:

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

- Second order prediction:

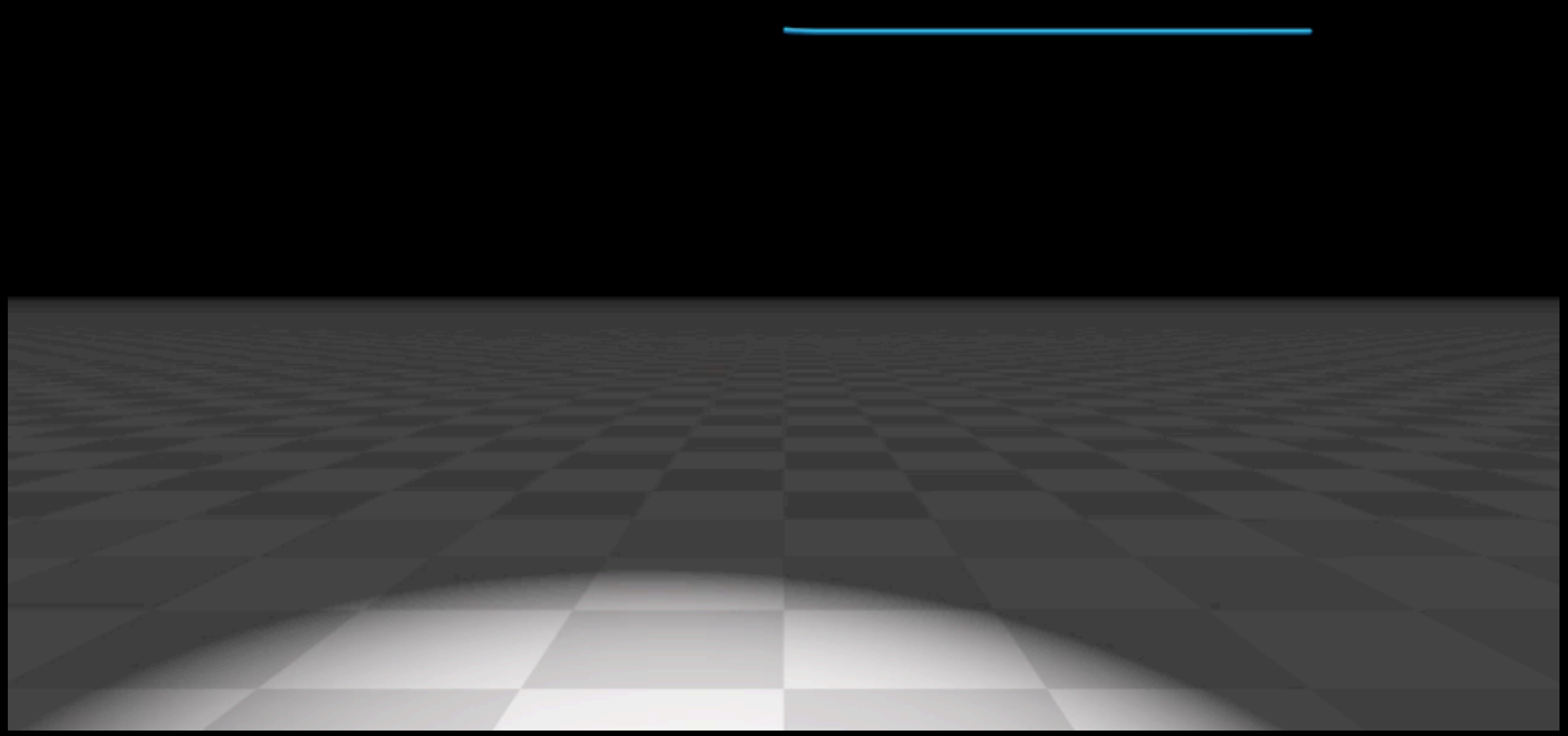
$$\tilde{\mathbf{x}} = \frac{4}{3}\mathbf{x}^n - \frac{1}{3}\mathbf{x}^{n-1} + \frac{8}{9}\Delta t \mathbf{v}^n - \frac{2}{9}\Delta t \mathbf{v}^{n-1} + \frac{4}{9}\Delta t^2 \mathbf{M}^{-1}\mathbf{f}_{ext}$$

- Second order velocity update:

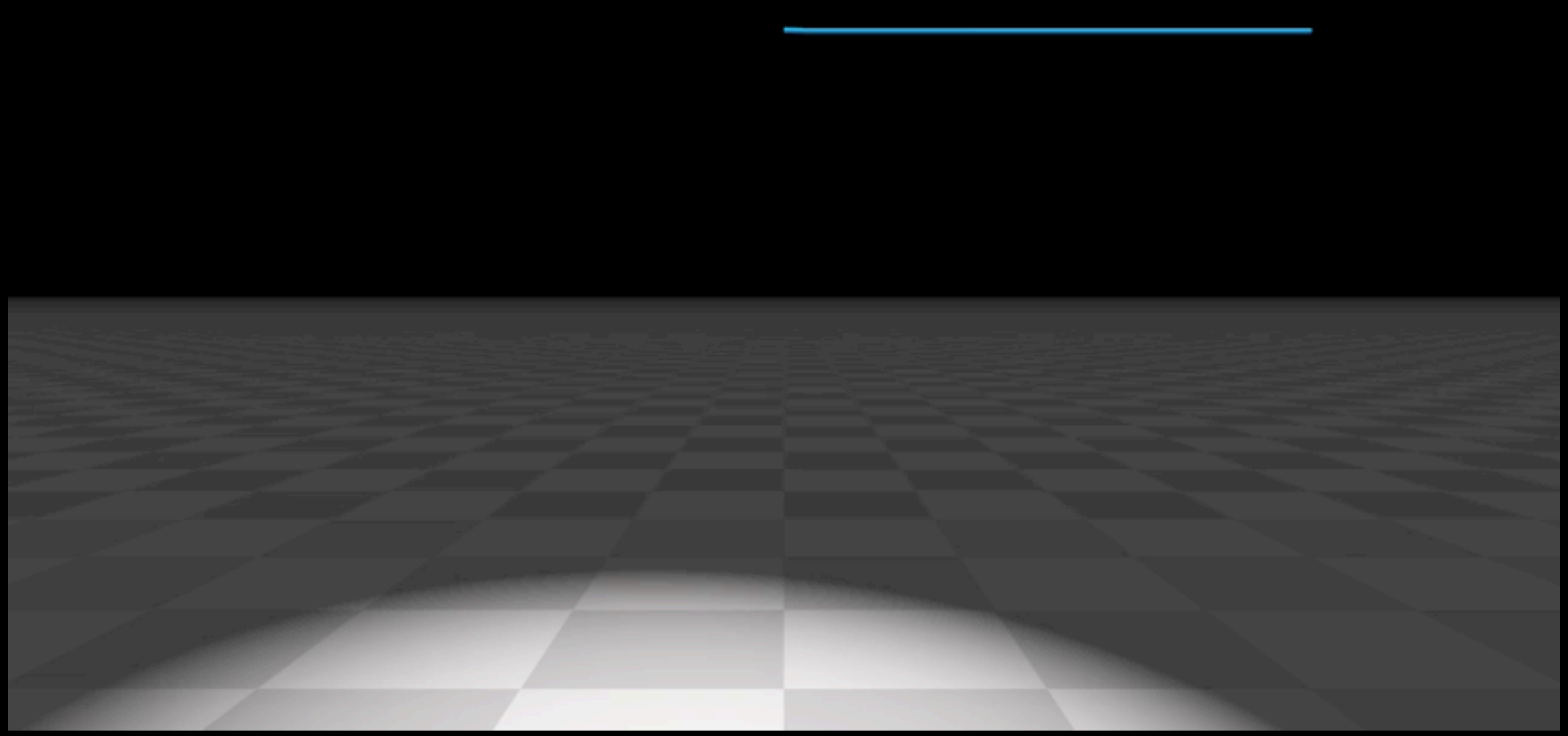
$$\mathbf{v}^{n+1} = \frac{1}{\Delta t}\left[\frac{3}{2}\mathbf{x}^{n+1} - 2\mathbf{x}^n + \frac{1}{2}\mathbf{x}^{n-1}\right].$$

- See [English 08]

Second Order PBD

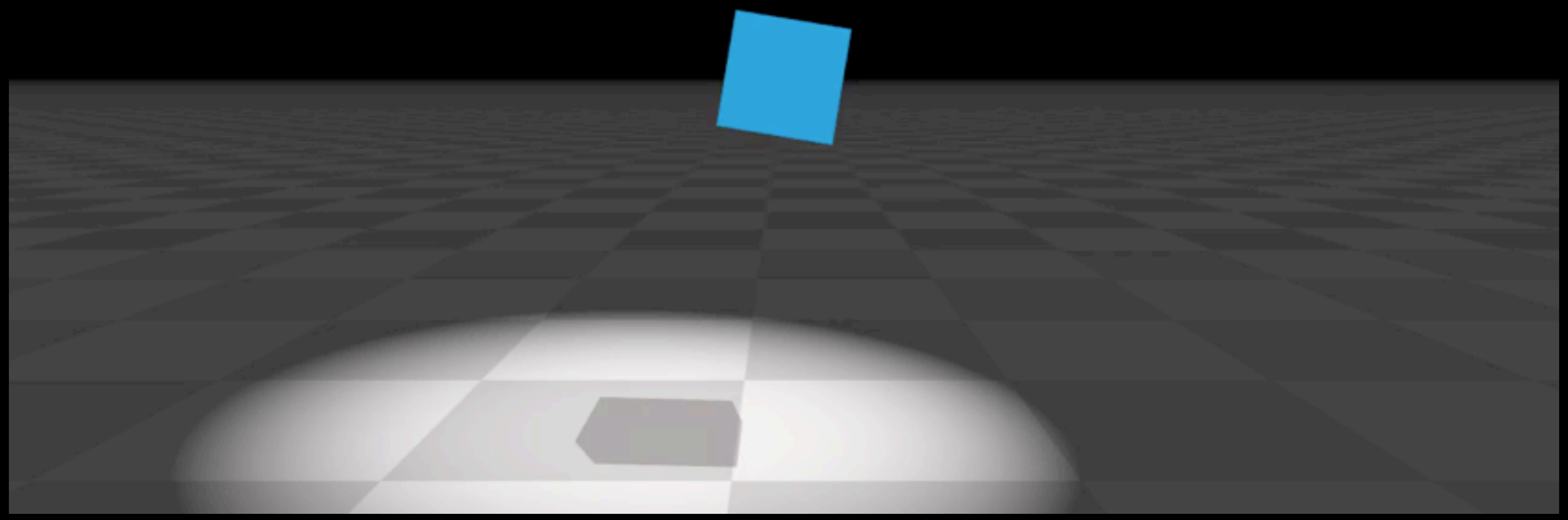


First Order

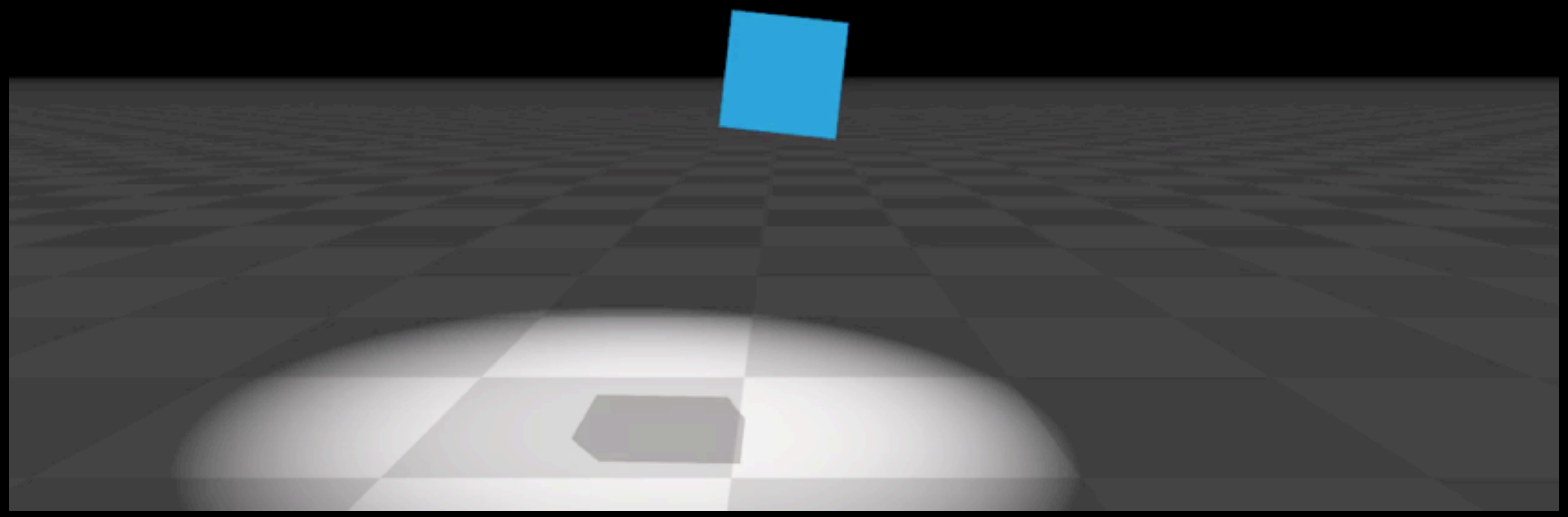


Second Order

Second Order PBD

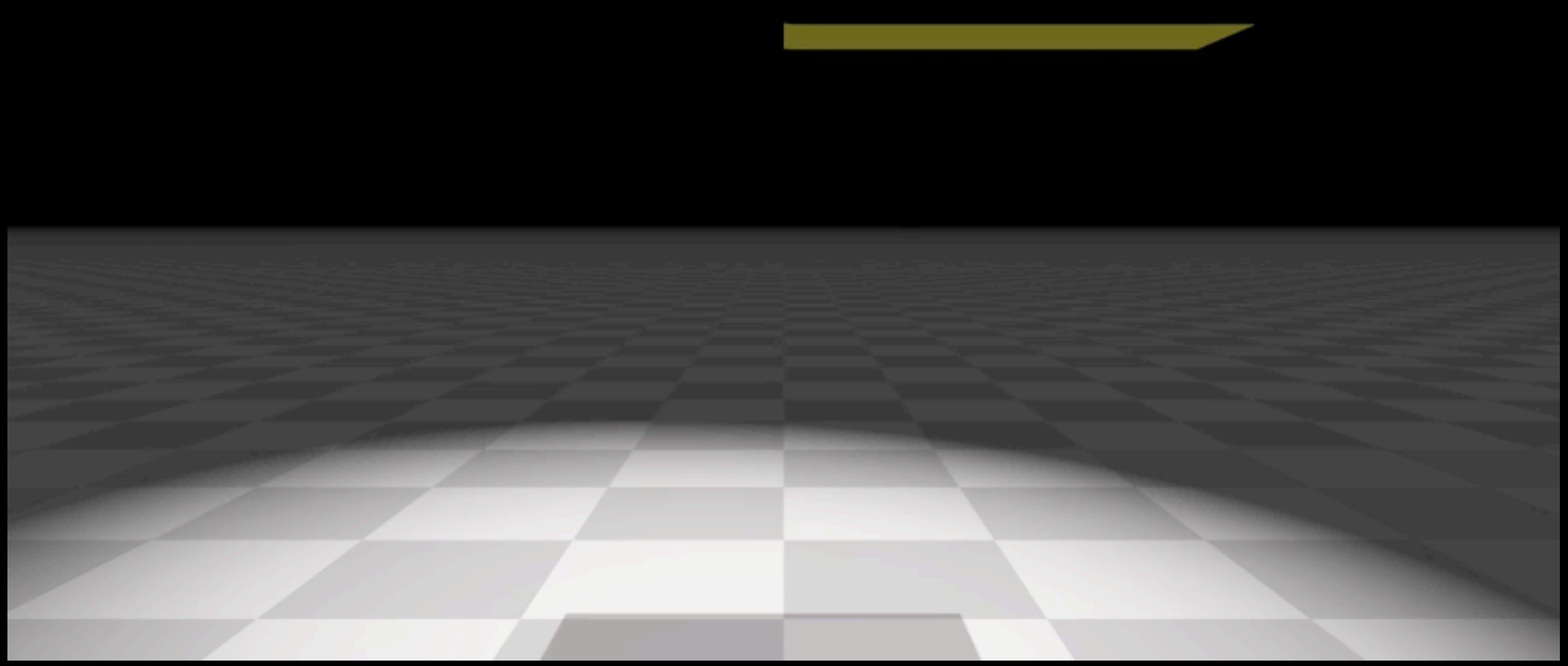


First Order

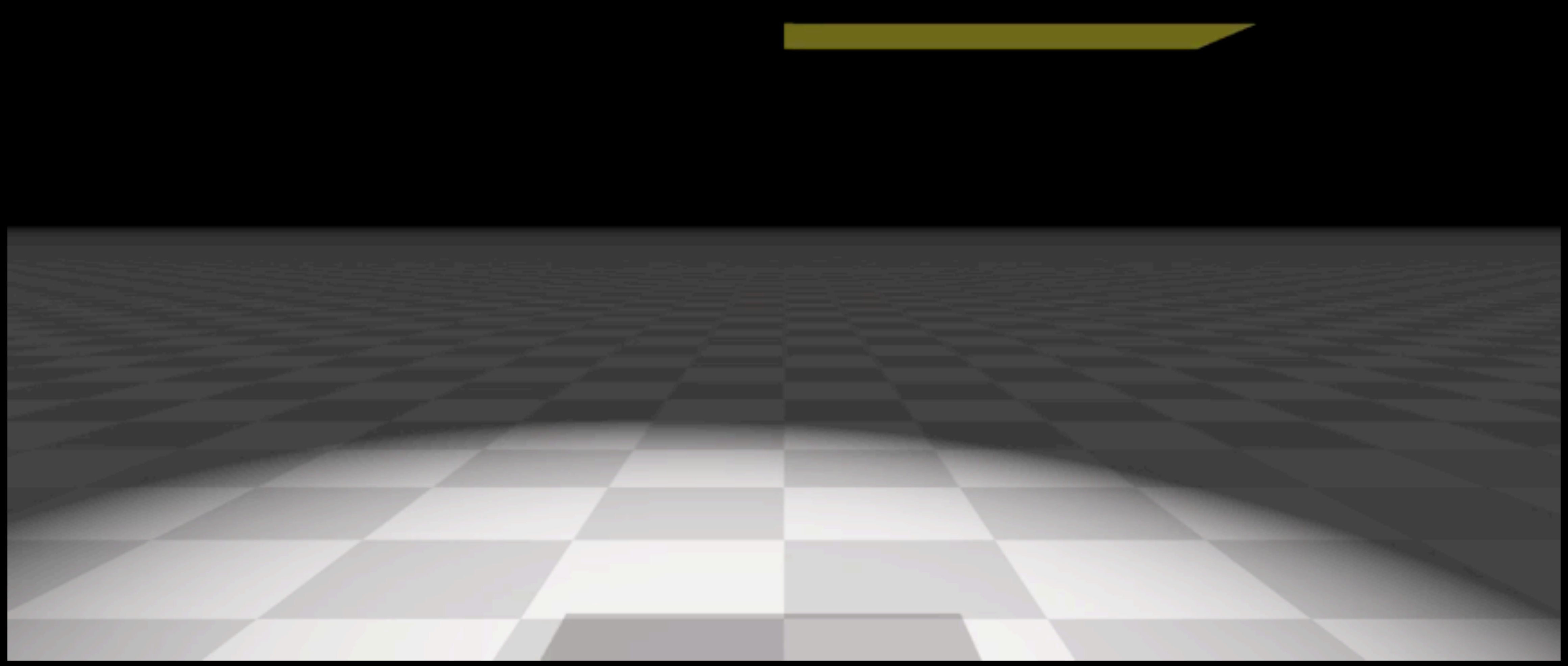


Second Order

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

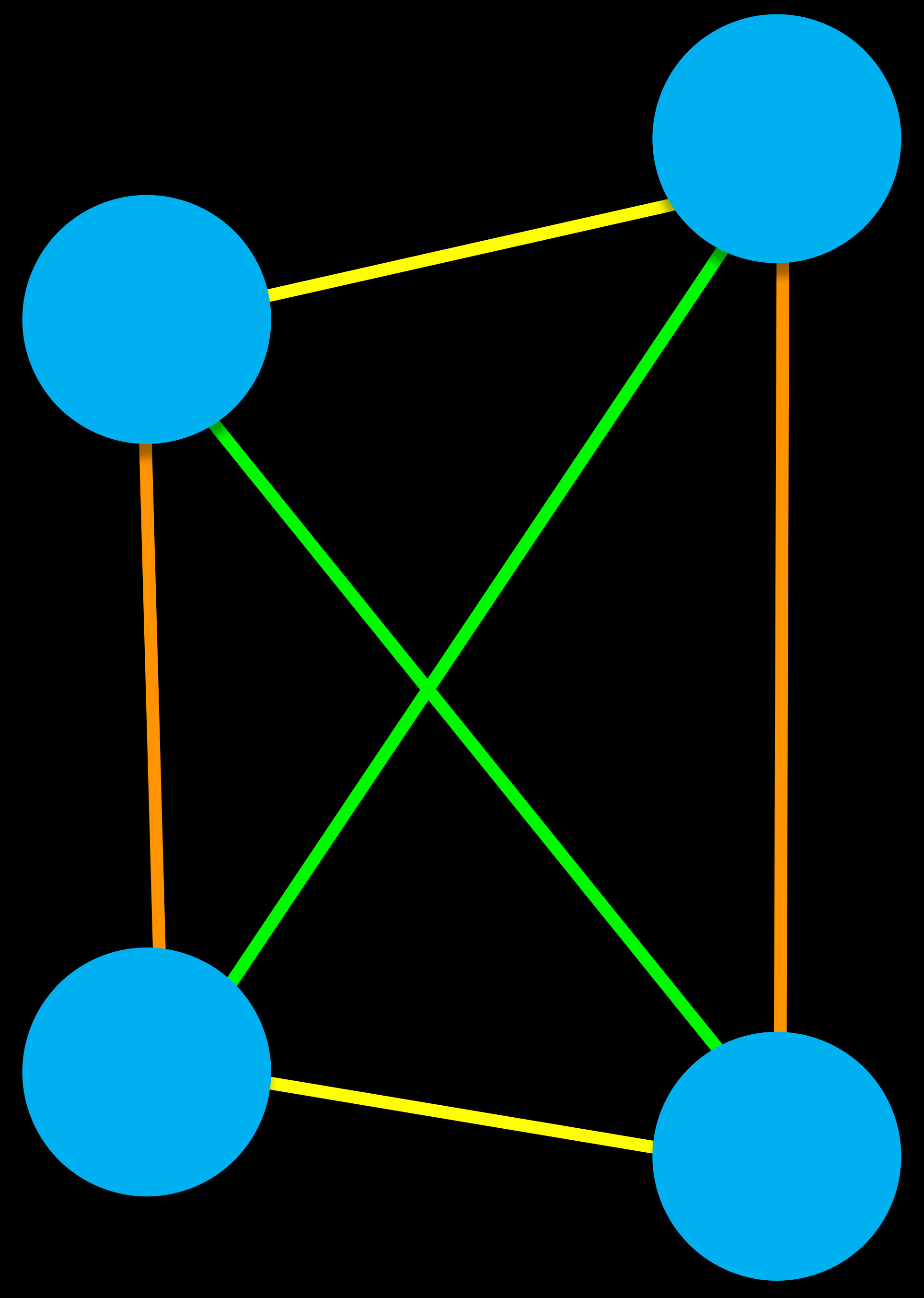
Implementation

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 or particle in parallel
- 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)`
	`  }`
	`}`

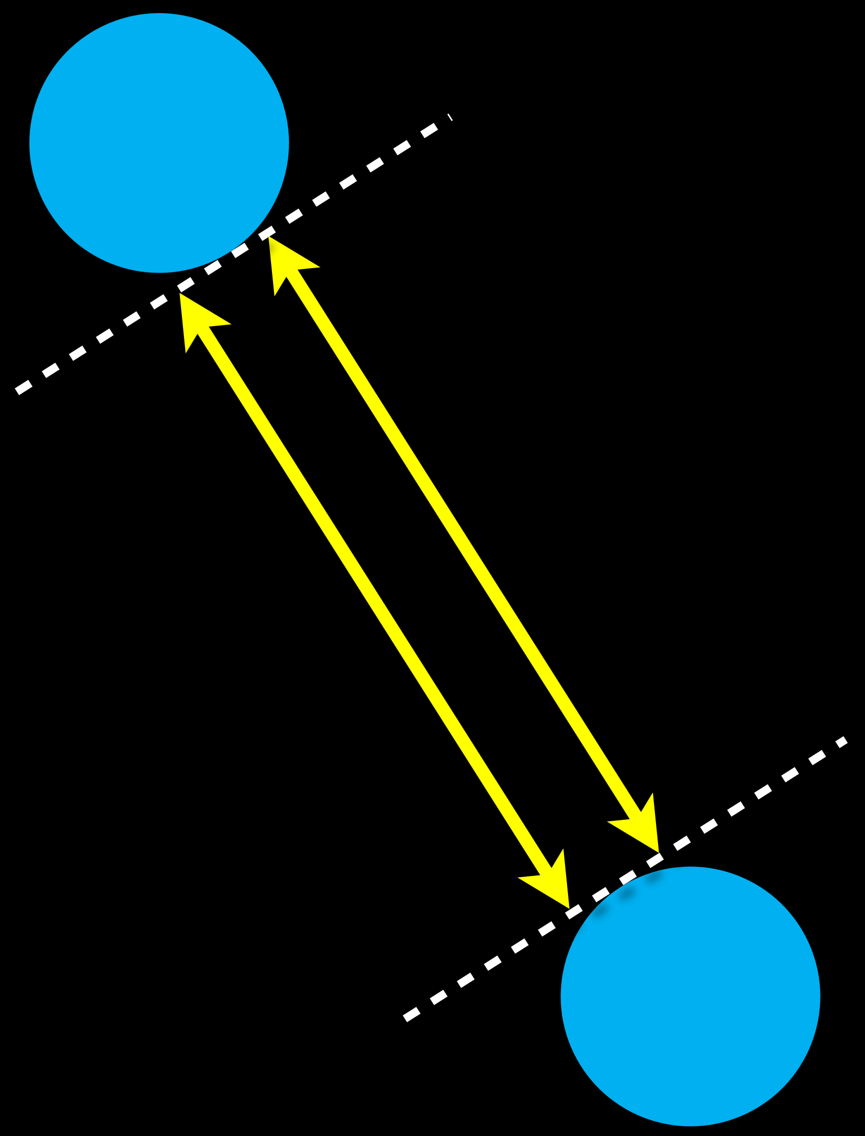
Jacobi Methods

- 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]:

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



Parallel Methods Comparison

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

NVIDIA

Hybrid Parallel Methods

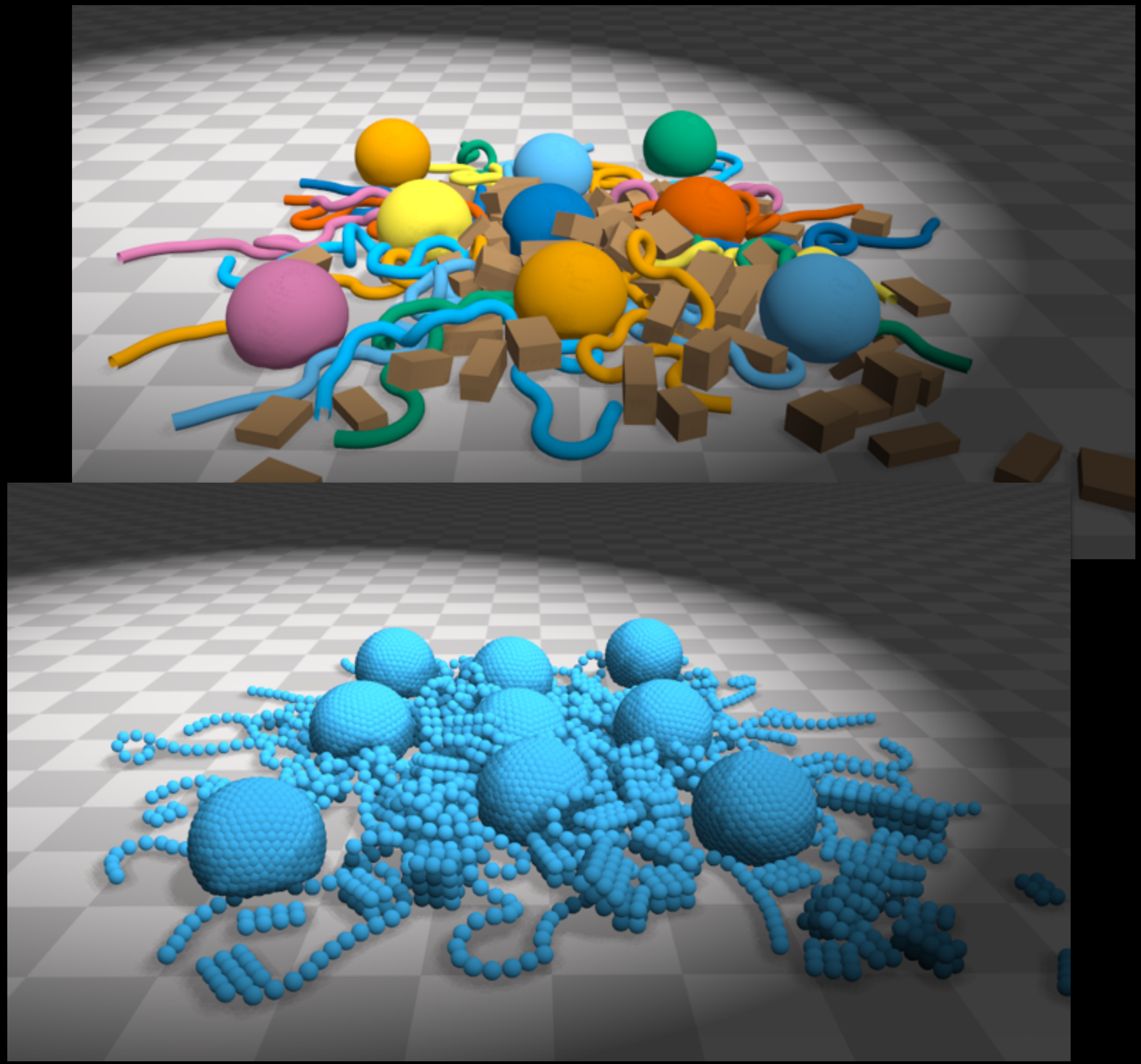
- Best of both worlds
- Perform graph-coloring
- Upper limit on number of colors
- Process everything else with Jacobi
- [Fratarcangeli & Pellacini 2015]

Solver Framework

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



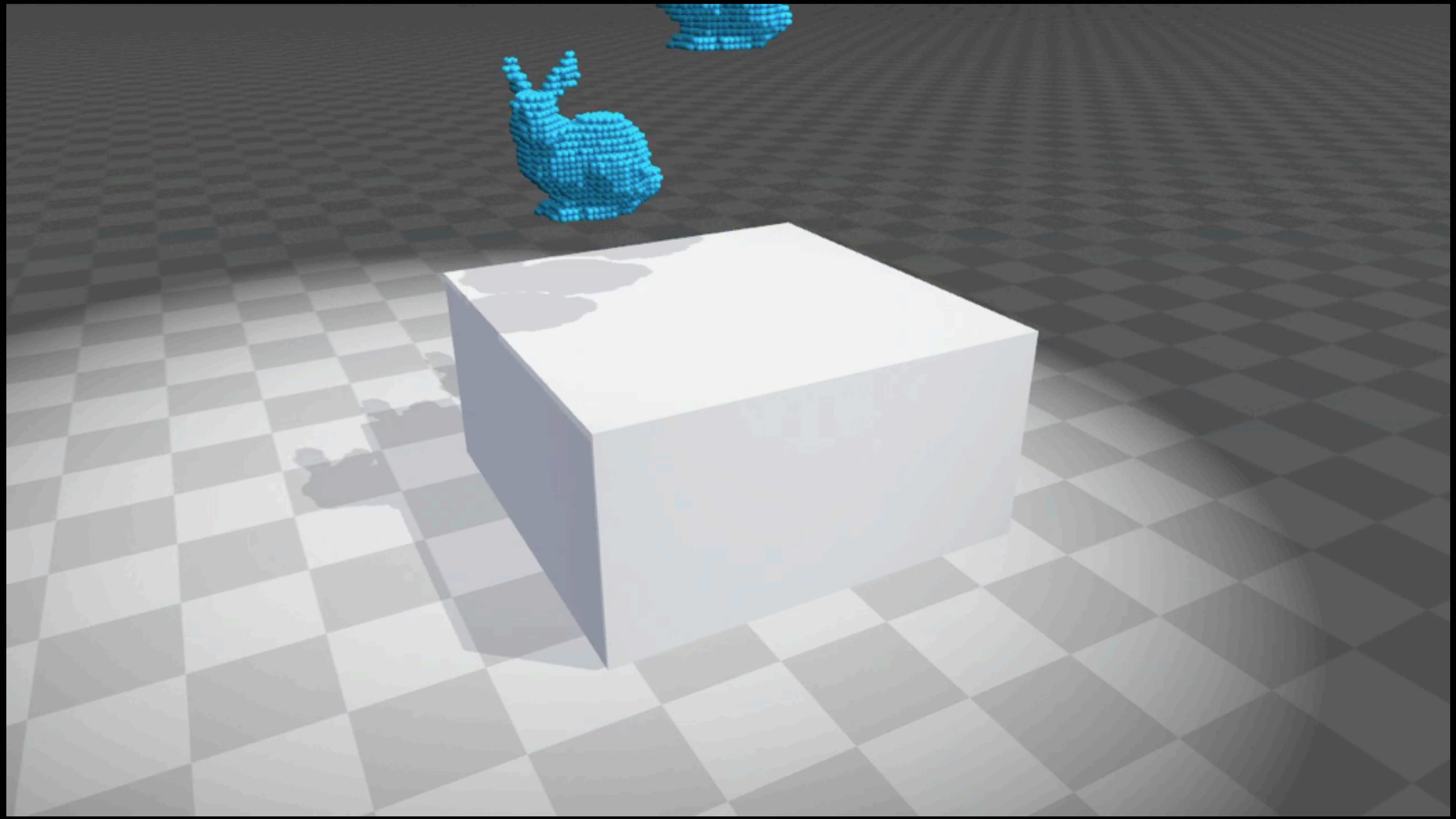
Particles

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

- Velocity stored explicitly
- Phase-ID used to control collision filtering
- Global radius
- SOA layout

Constraints

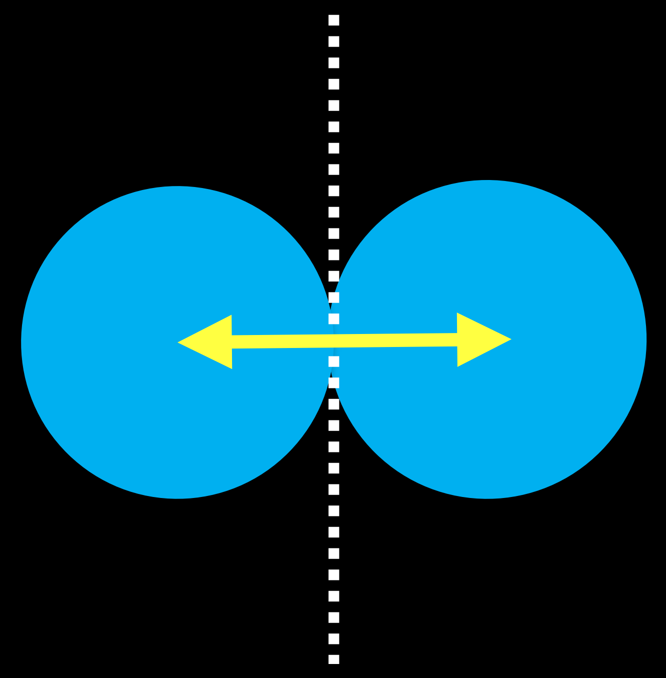
- Constraint types:
  - Distance (clothing)
  - Shape (rigids, plastics)
  - Density (fluids)
  - Volume (inflatables)
  - Contact (non-penetration)
- Combine constraints
  - Melting, phase-changes
  - Stiff cloth, bent metal



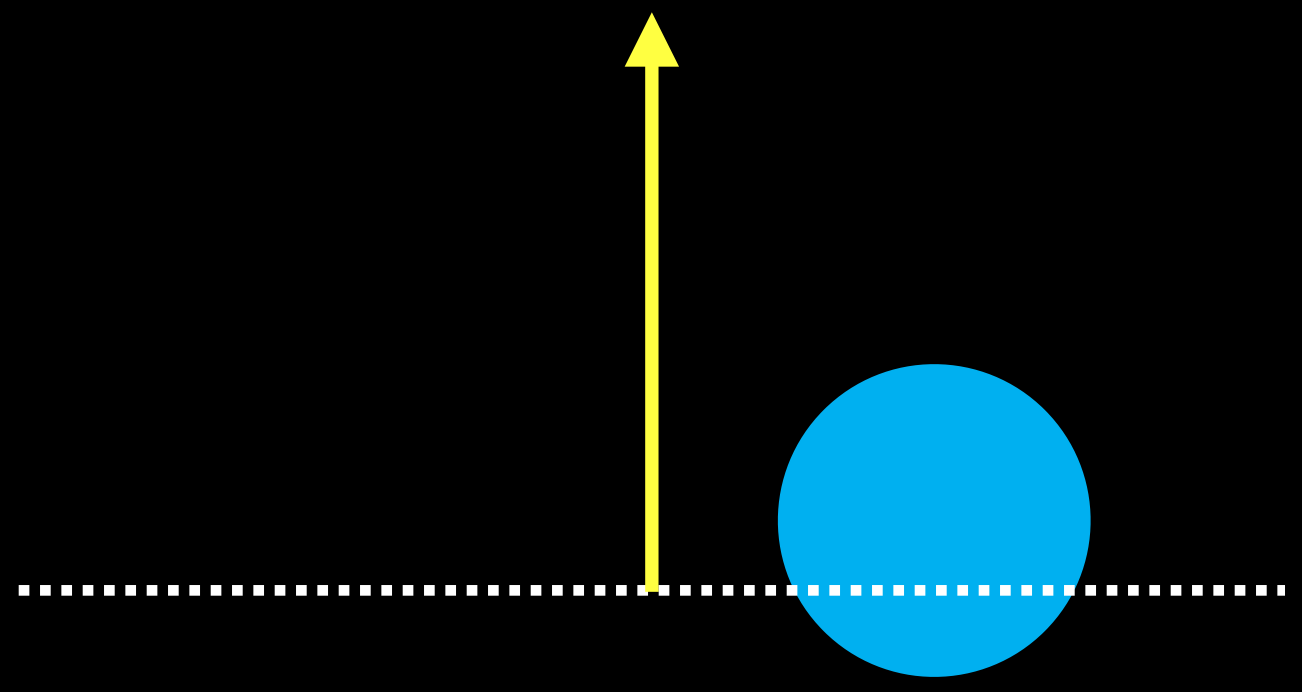
Contact and Friction

Collision Detection Between Particles

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



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

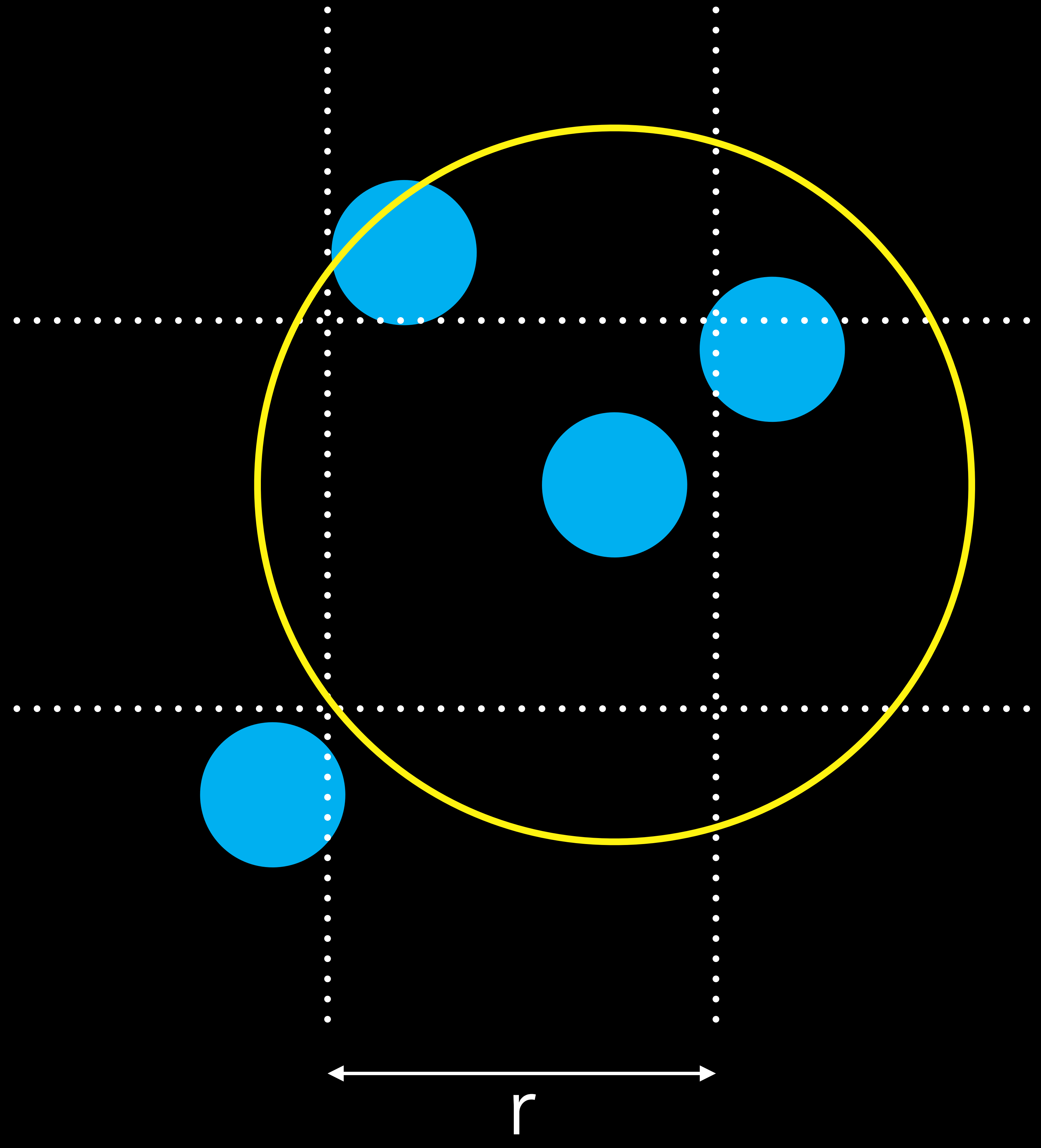


$$C_{contact} = \mathbf{n} \cdot \mathbf{x} - r \geq 0$$

NVIDIA

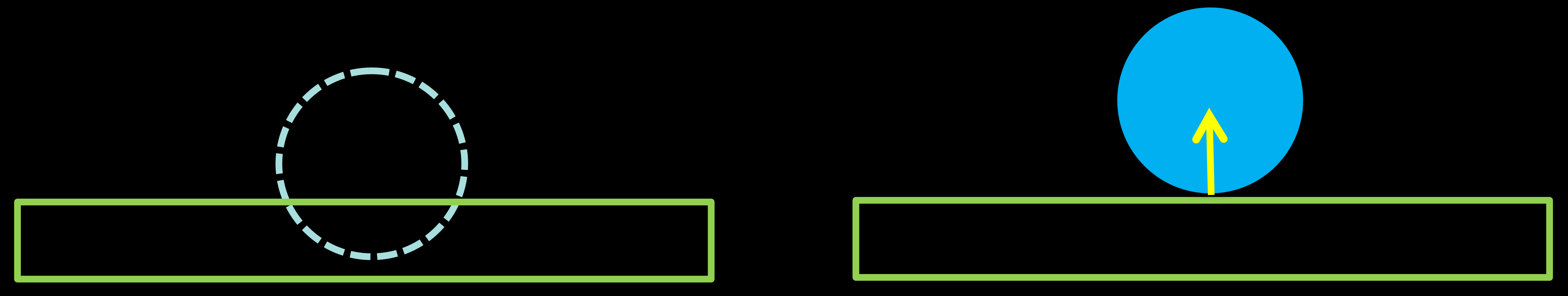
Collision Detection Between Particles

- 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]



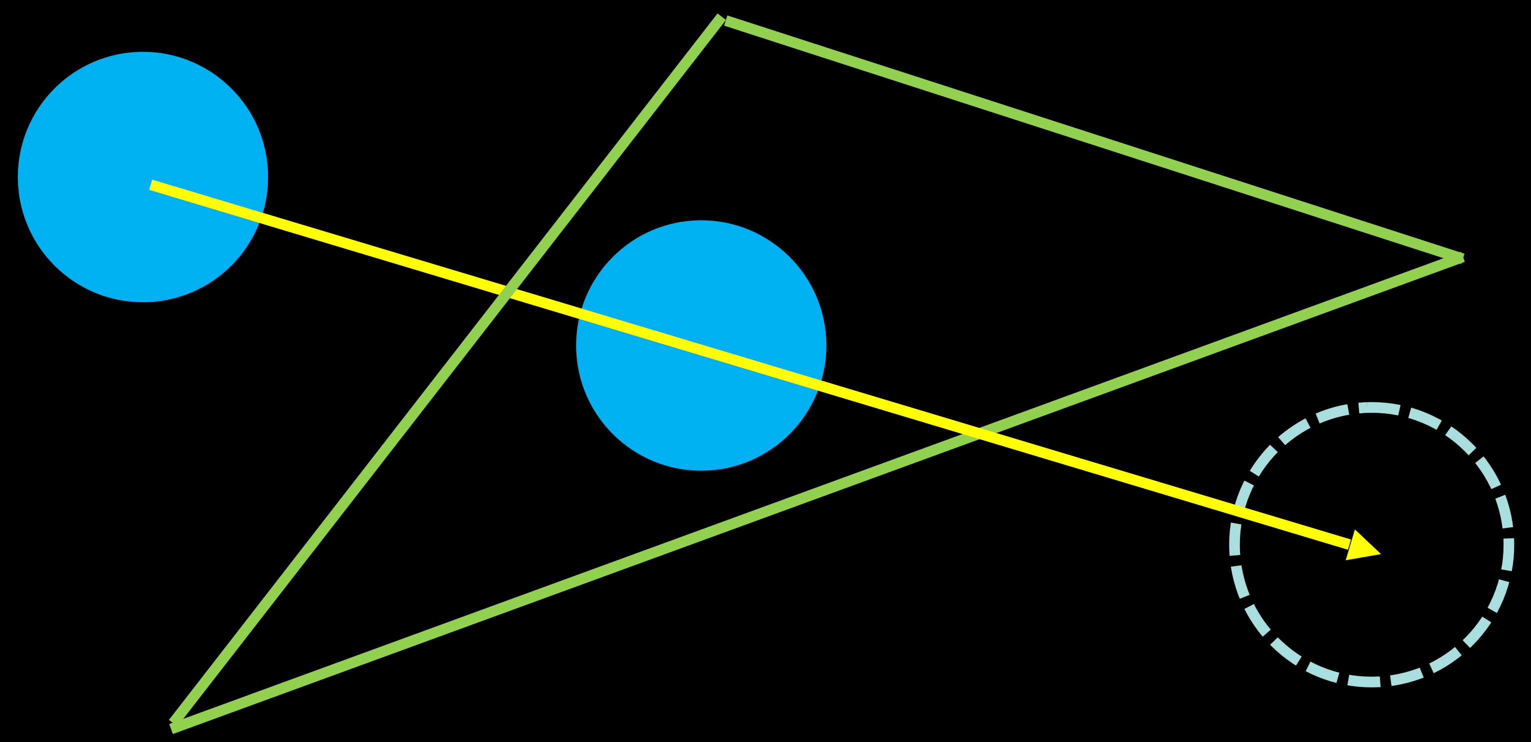
Collision Detection Against Shapes

- Particle-Convex
  - 2D hash-grid
  - Built on GPU



Convex Collision (MTD)

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



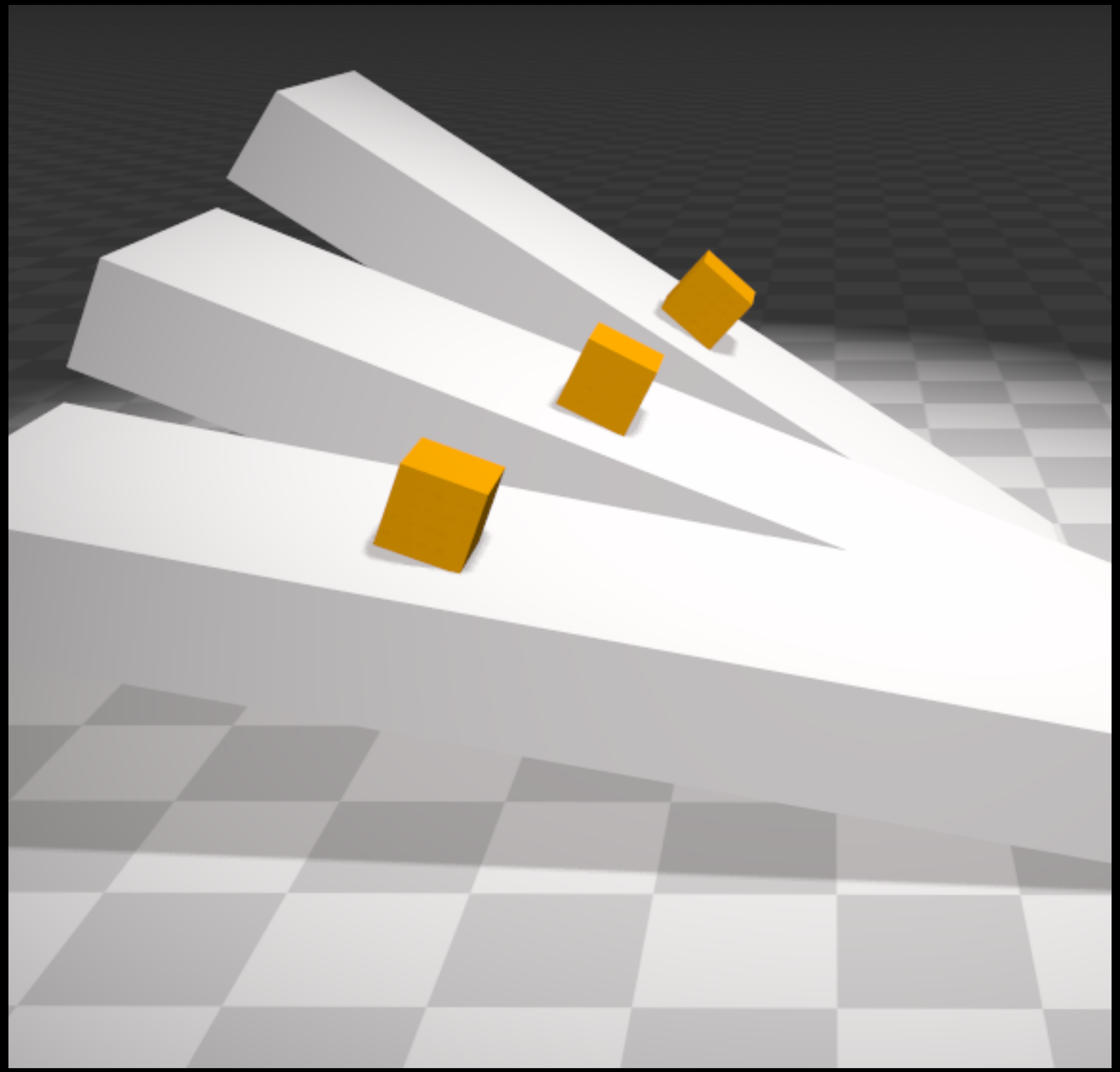
Triangle Collision (TOI)

Friction

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

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

- Approximate Coulomb friction using penetration depth to limit constraint lambda
- Generates convincing particle piling
- [Francu 2017]

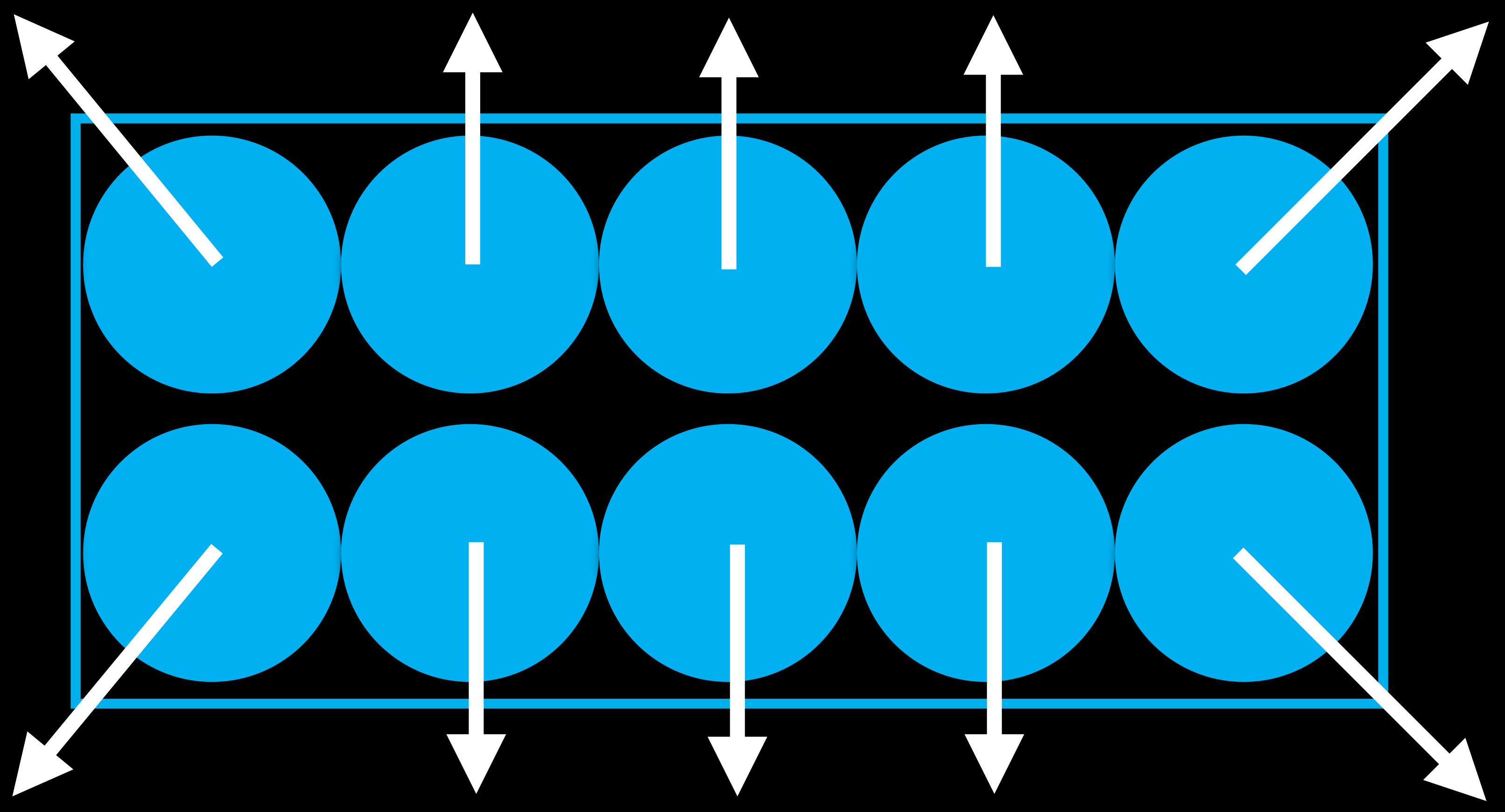


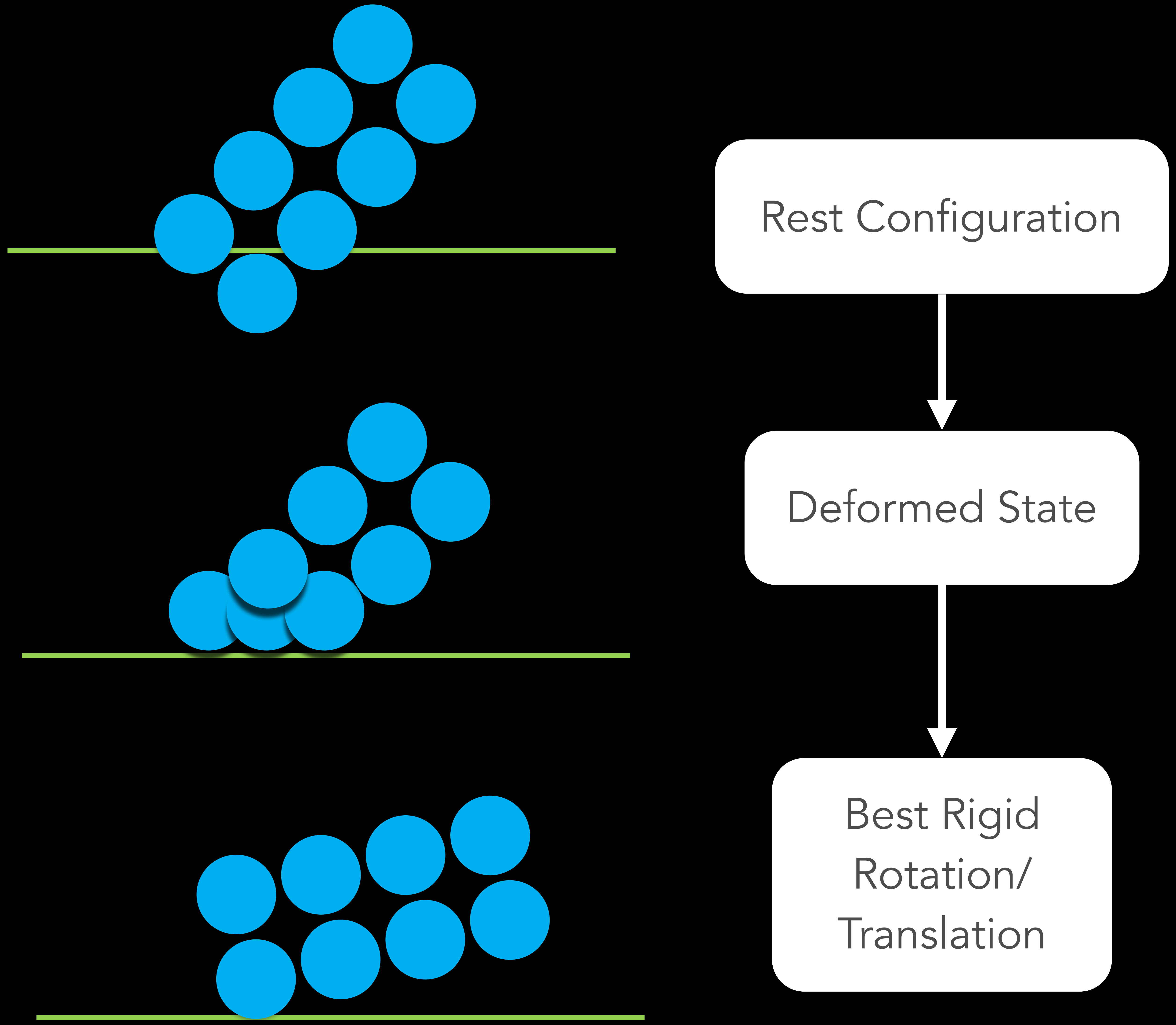
NVIDIA

Rigid Bodies

Rigid Bodies

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

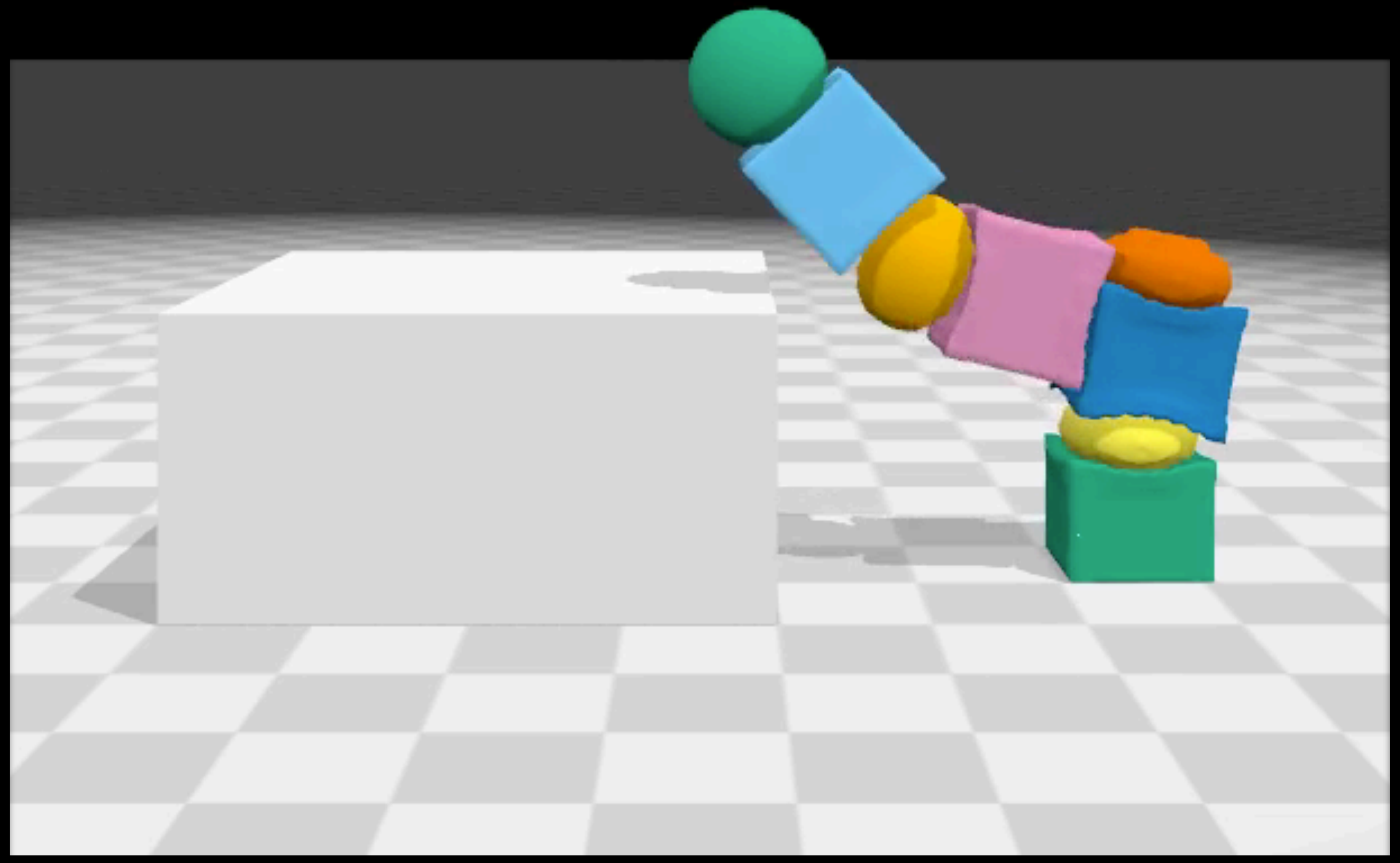




NVIDIA

Plastic Deformation

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



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

Robust and Simple Polar Decomposition

- Shape matching requires a polar decomposition
- Can be done through SVD / Eigenvalue decomposition
- Complex code, ill-posed for indefinite systems
- Simple algorithm given in [Müller et al 2016]
- Robustly handles inversion through temporal coherence

```
void extractRotation(const Matrix3d &A, Quaterniond &q,
     const unsigned int maxIter)
   for (unsigned int iter = 0; iter < maxIter; iter++)
   {
      Matrix3d R = q.matrix();
      Vector3d omega = (R.col(0).cross(A.col(0)) + R.col
           (1).cross(A.col(1)) + R.col(2).cross(A.col(2))
           ) * (1.0 / fabs(R.col(0).dot(A.col(0)) + R.col
           (1).dot(A.col(1)) + R.col(2).dot(A.col(2))) +
           1.0e-9);
      double w = omega.norm();
      if (w < 1.0e-9)
         break;

      q = Quaterniond(AngleAxisd(w, (1.0/w)*omega)) * q;
      q.normalize();
   }
}
```


Scene
Soft Octopus
Soft Teapot
Soft Rope
Soft Cloth
Soft Bowl
Soft Rod
Soft Armadillo
Soft Bunny
Mixed Pile
Options
Global
Emit particles
Pause
Wireframe
Draw Points
Draw Fluid
Draw Mesh
Draw Basis
Draw Springs
Reset Scene
Num Substeps 2
Num Iterations 4
Gravity X 0
Frame: 0
Particle Count: 4389
Diffuse Count: 0
Rigid Count: 270
Spring Count: 0
Num Substeps: 2
Num Iterations: 4
CUDA Device: GeForce GTX TITAN X

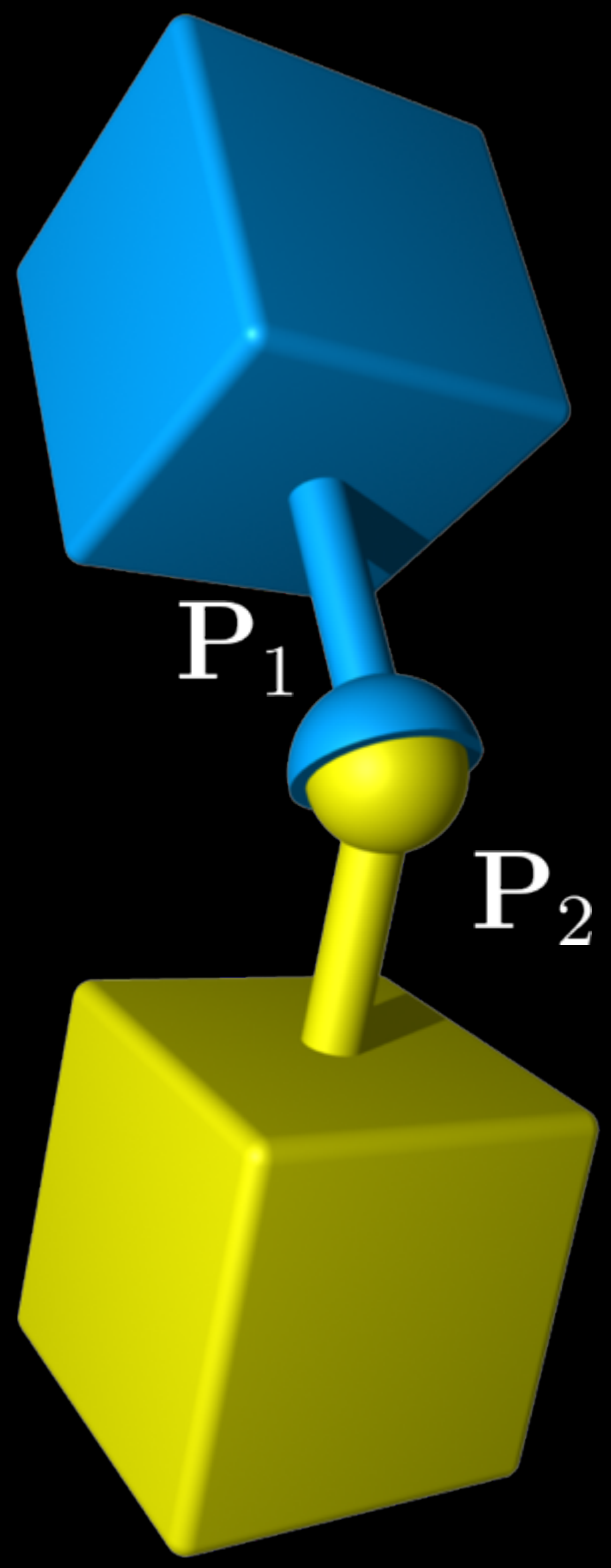
Generalised Coordinate Rigid Bodies

- Particle: $\mathbf{P}(\mathbf{x}) = \mathbf{x}$
- Rigid body: $\mathbf{P}(\mathbf{x}, \boldsymbol{\vartheta}) = \mathbf{x} + \mathbf{R}(\boldsymbol{\vartheta})\mathbf{P}_{\text{local}}$

- Rotation is parameterized by exponential map $\boldsymbol{\vartheta}$
- Example, ball joint:

$$\mathbf{C}(\mathbf{P}_1, \mathbf{P}_2) = \mathbf{P}_1 - \mathbf{P}_2 = \mathbf{0}$$

- [Deul et al. 2014]



Generalized Rigid Body Constraint Gradients

- Split gradient into a constraint part and connector part
- Particle:

$$\nabla \mathbf{C} = \underbrace{\frac{\partial \mathbf{C}(\mathbf{P})}{\partial \mathbf{P}}}_{\substack{\text{constraint} \\ \text{specific part}}} \cdot \underbrace{\frac{\partial \mathbf{P}}{\partial \mathbf{x}}}_{\substack{\text{connector} \\ \text{specific part}}} = \frac{\partial \mathbf{C}(\mathbf{P})}{\partial \mathbf{P}}$$

- Rigid Body:

$$\nabla \mathbf{C} = \underbrace{\frac{\partial \mathbf{C}(\mathbf{P})}{\partial \mathbf{P}}}_{\substack{\text{constraint} \\ \text{specific part}}} \cdot \underbrace{\begin{pmatrix} \frac{\partial \mathbf{P}}{\partial \mathbf{x}} & \frac{\partial \mathbf{P}}{\partial \boldsymbol{\vartheta}} \end{pmatrix}^T}_{\substack{\text{connector} \\ \text{specific part}}}$$

NVIDIA

Generalised Position-Based Solver

- Linearization of constraint (rigid bodies):

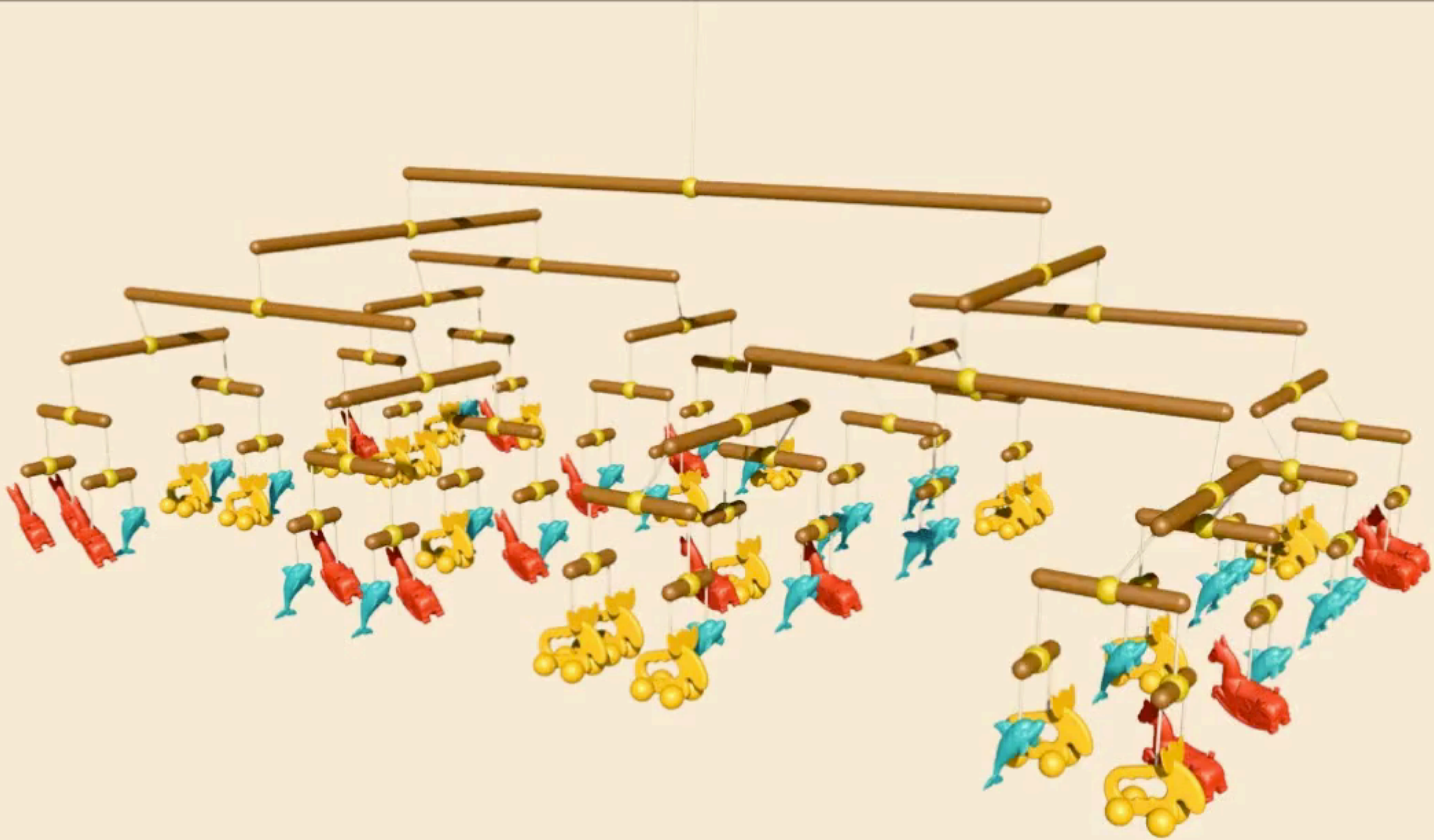
$$\mathbf{C}(\mathbf{x} + \Delta\mathbf{x}, \boldsymbol{\varphi} + \Delta\boldsymbol{\varphi}) \approx \mathbf{C}(\mathbf{x}, \boldsymbol{\varphi}) + \nabla\mathbf{C}(\Delta\mathbf{x}^T, \Delta\boldsymbol{\varphi}^T)$$

- Computation of Lagrange multiplier:

$$[\nabla\mathbf{C}\mathbf{M}^{-1}\nabla\mathbf{C}^T]\Delta\boldsymbol{\lambda} = -\mathbf{C}(\mathbf{x}_i, \boldsymbol{\varphi}_i)$$

- Correction vectors:

$$[\Delta\mathbf{x}^T, \Delta\boldsymbol{\varphi}^T] = \mathbf{M}^{-1}\nabla\mathbf{C}^T\boldsymbol{\lambda}$$

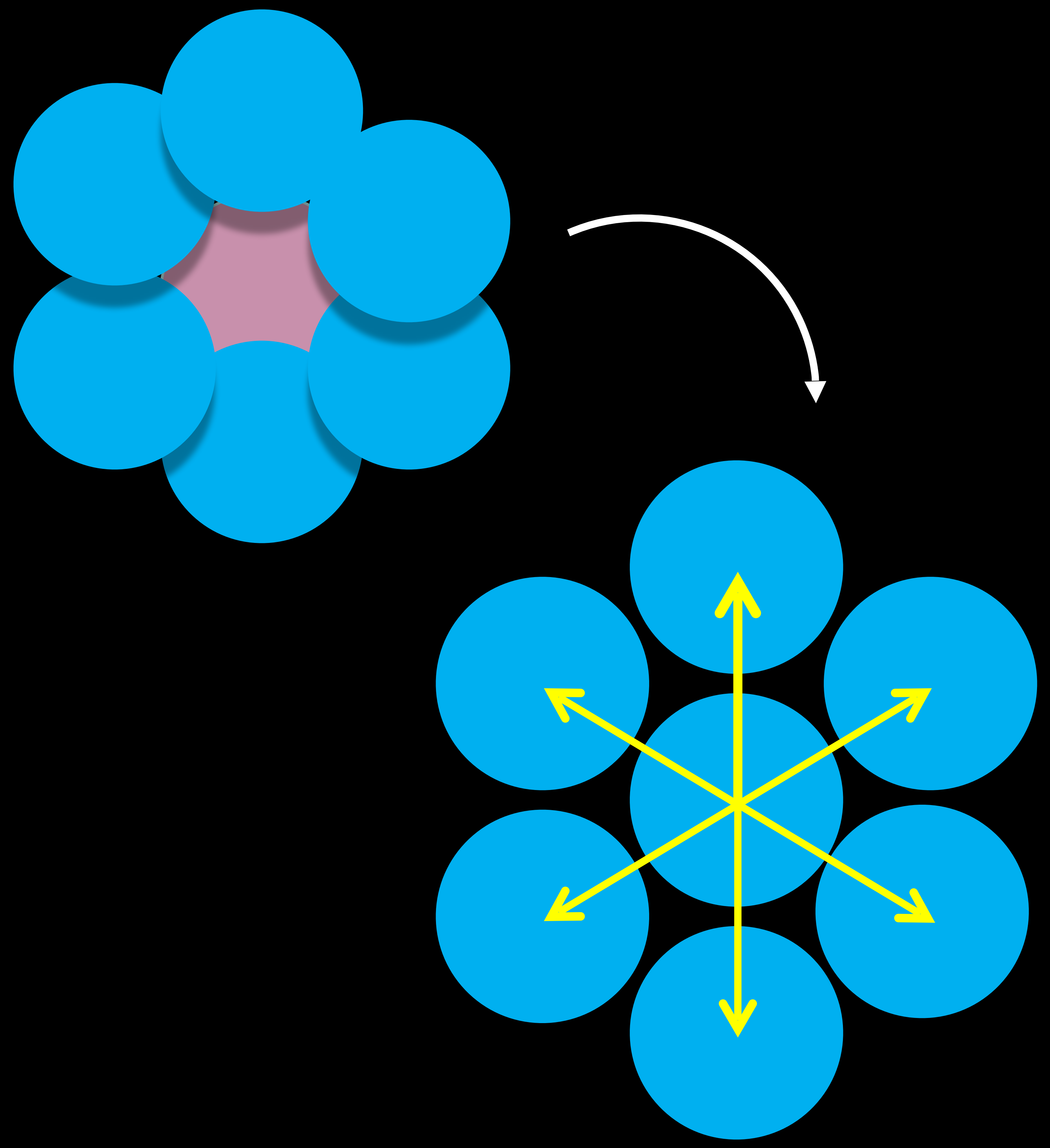


Fluids

Density Constraint

$$C_{density} = \frac{\rho_i}{\rho_0} - 1 \leq 0$$

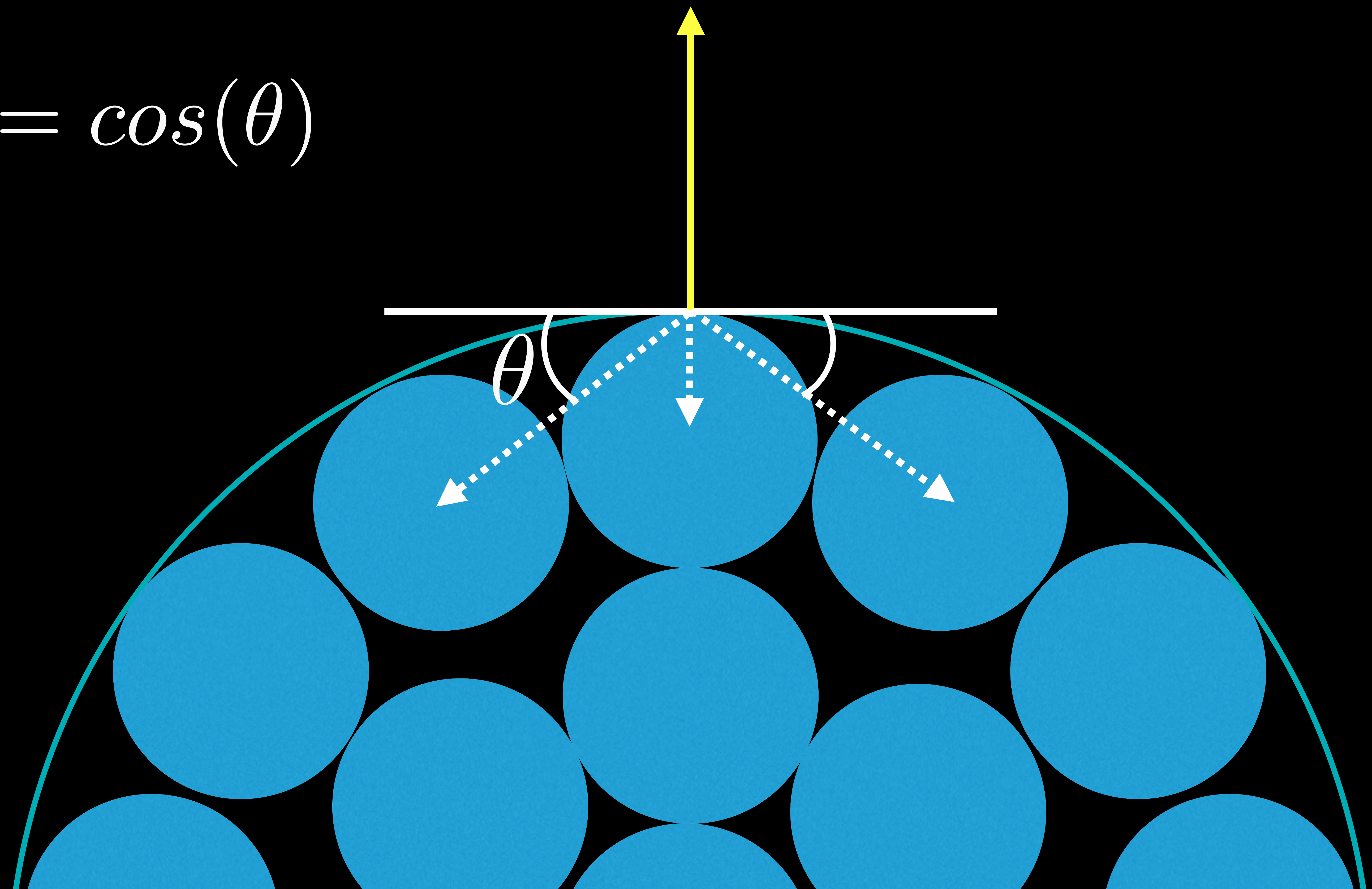
- Density via SPH kernels
- Unilateral constraint
- Cohesion from [Akinci13]



Surface Tension Constraint

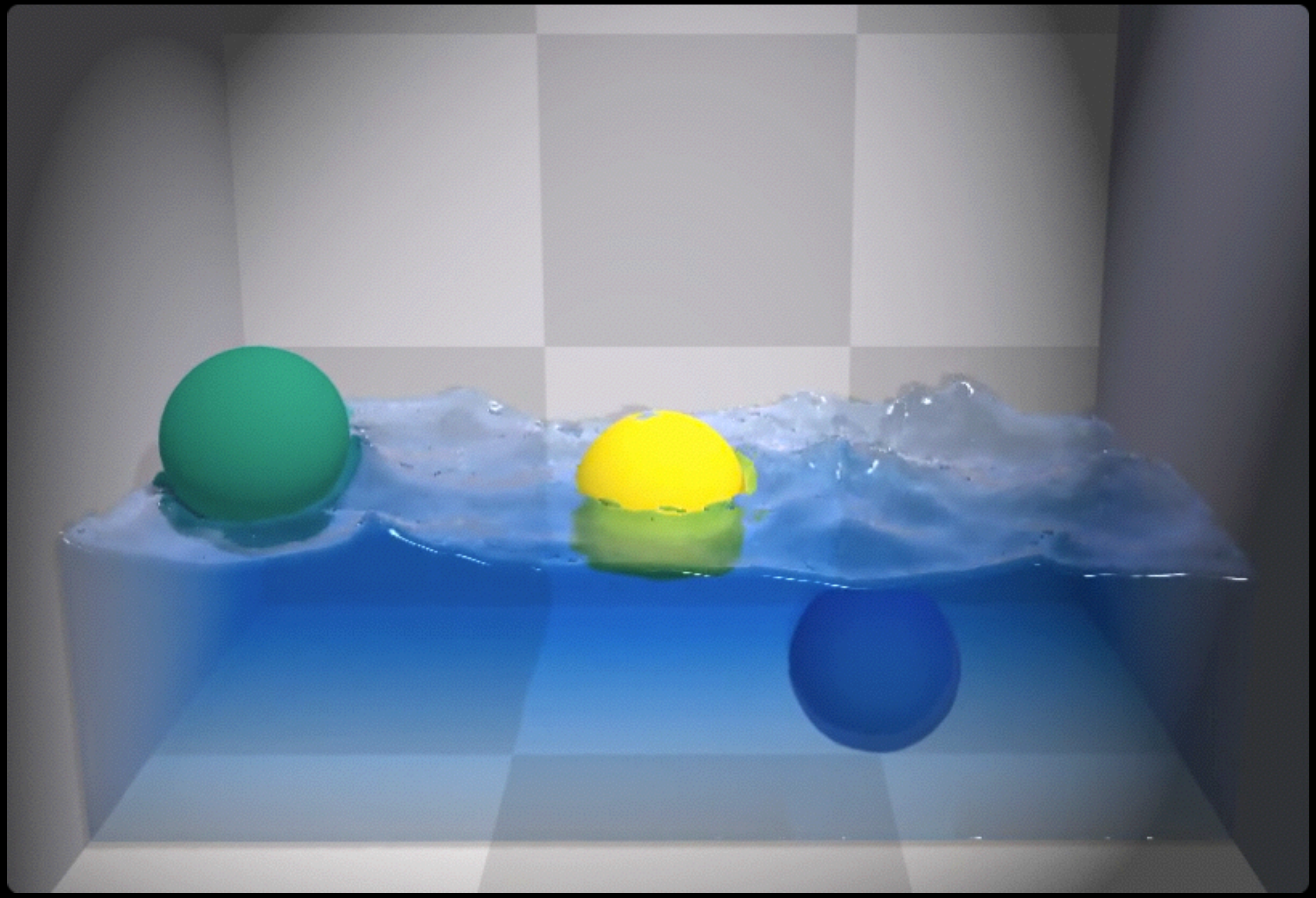
- Adapted surface tension model of [Akinci et al. 2013] to PBD
- Attempts to minimize curvature

$$C_{tension} = \bar{\mathbf{x_{ij}}} \cdot \bar{\mathbf{n_i}} = cos(\theta)$$



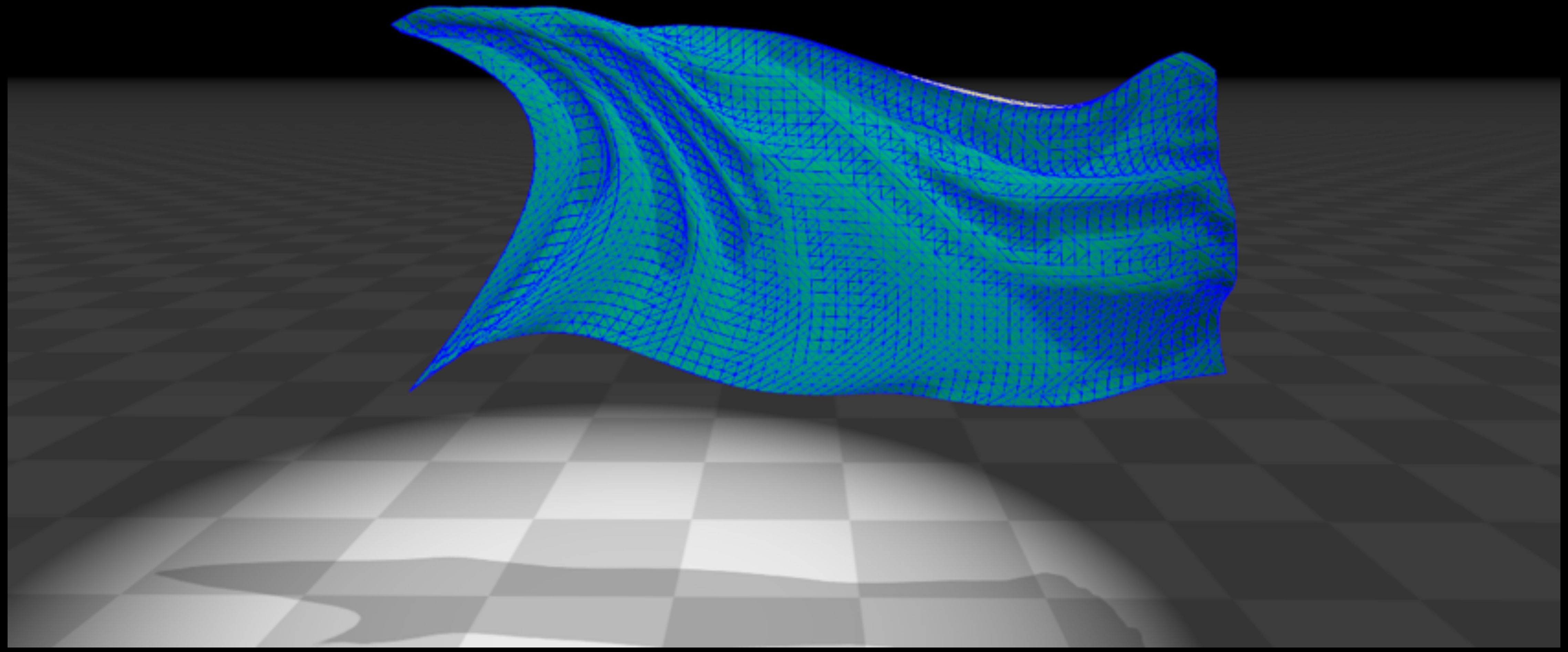
Two-Way Rigid Fluid Coupling

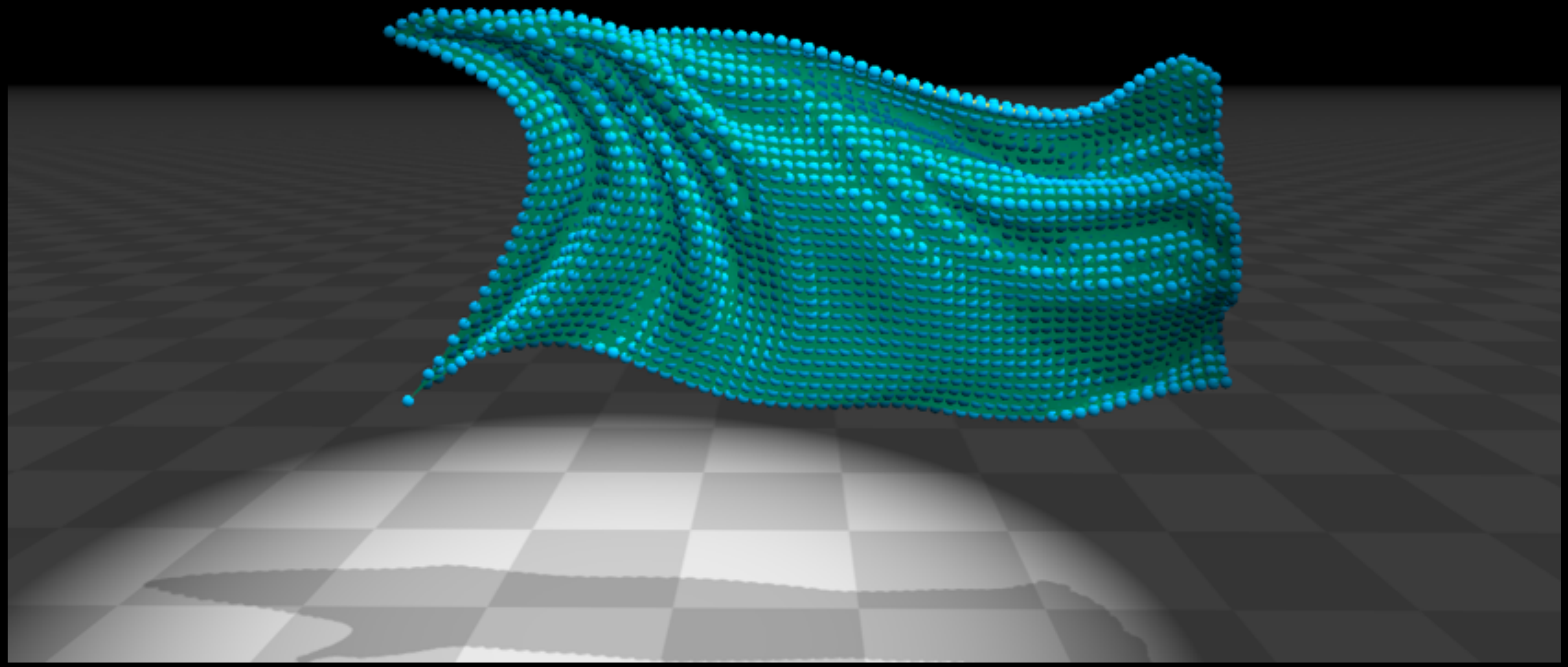
- Mostly automatic
- Include all particles in fluid density estimation
- Treat fluid->solid particle interactions as if both particles solid



Cloth

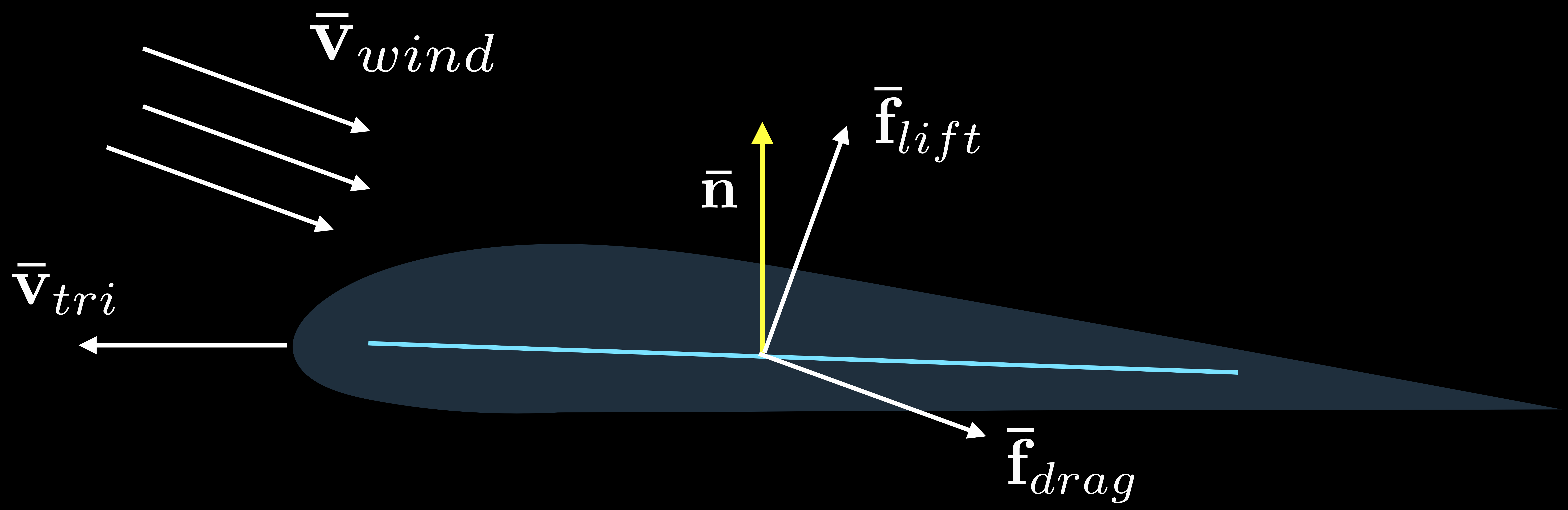
- Graph of distance + tether constraints
- Self-collision / inter-collision automatically handled





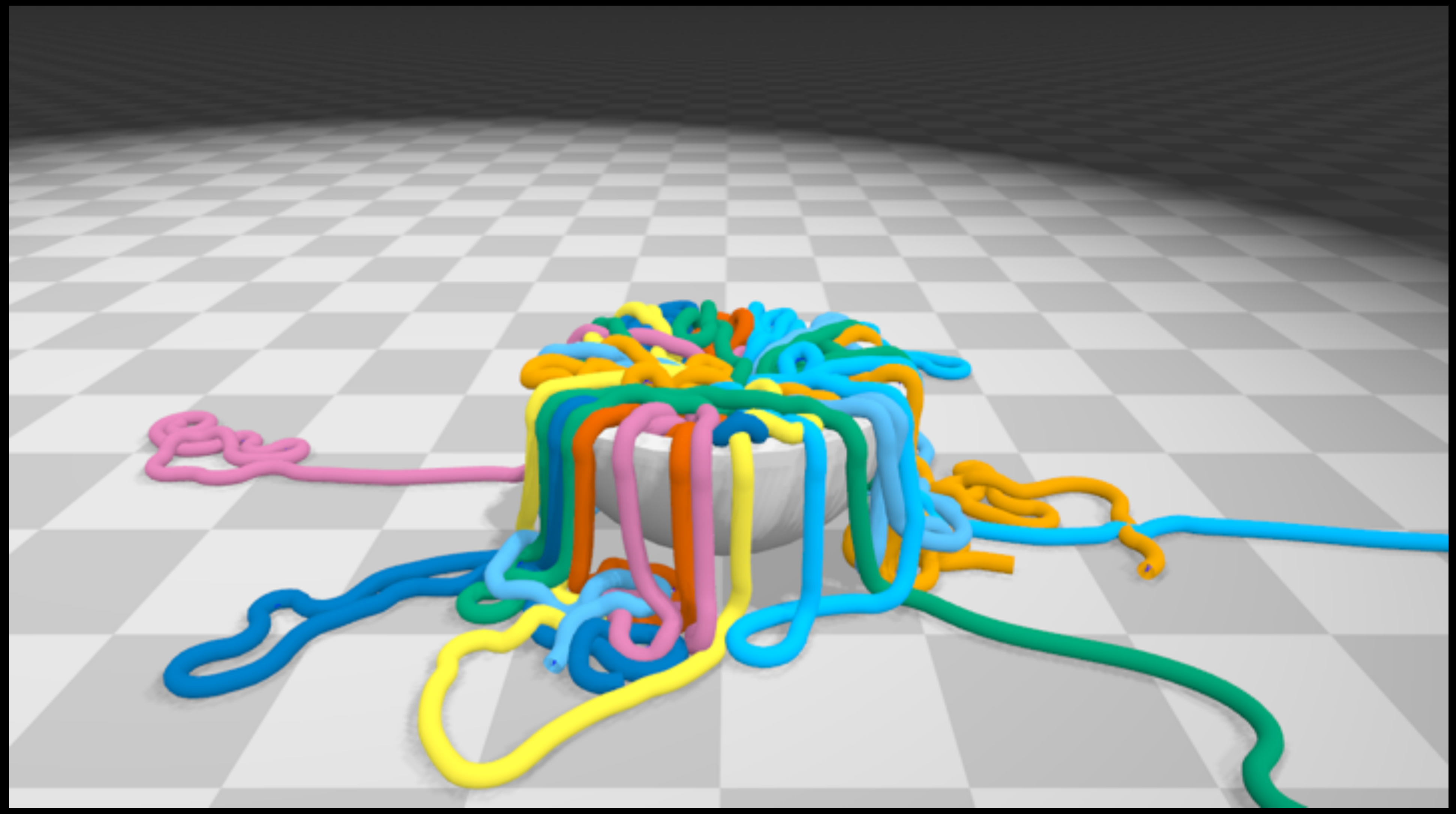
Cloth - Forces

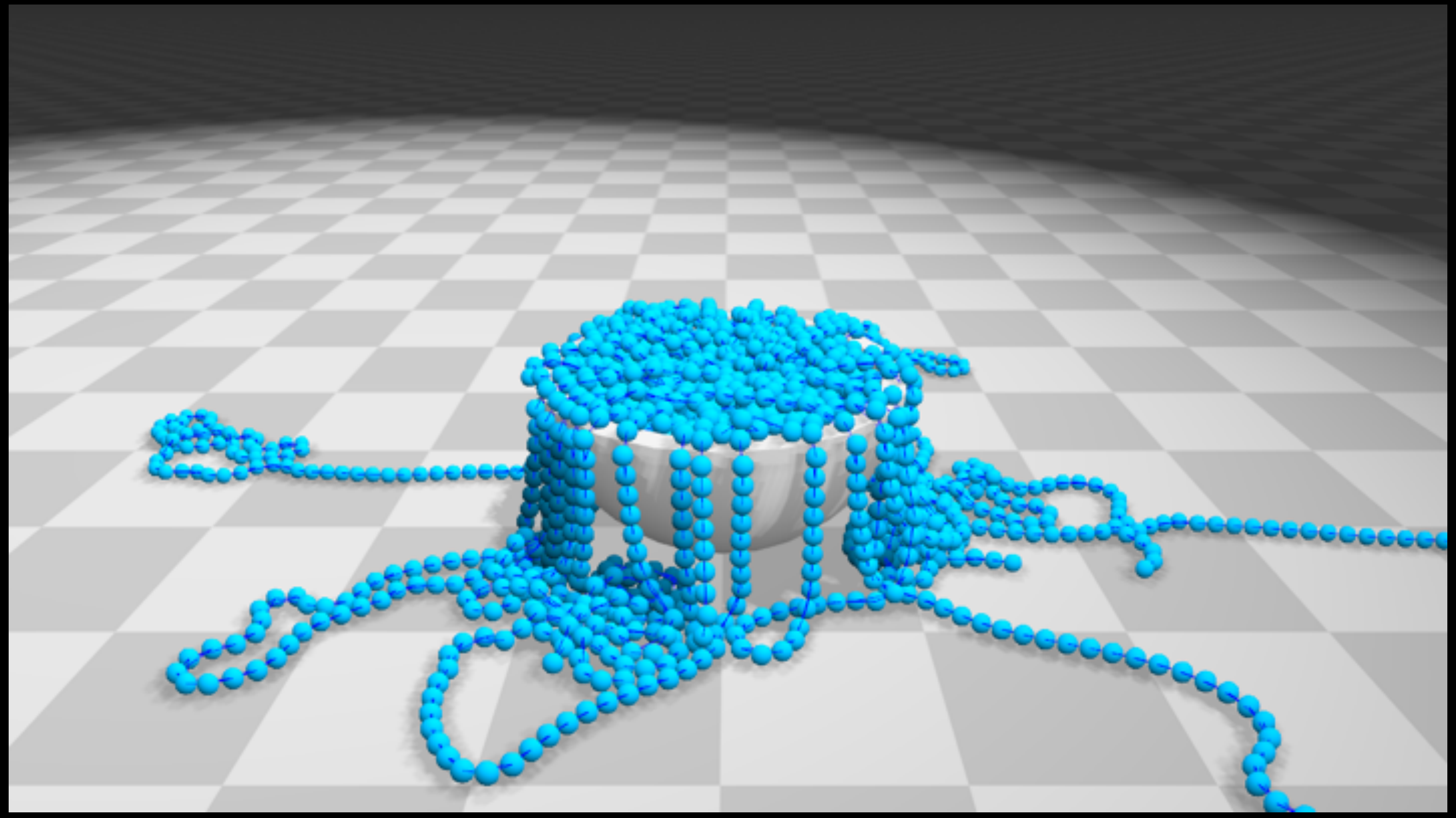
- Basic aerodynamic model
- Treat each triangle as a thin airfoil to generate lift + drag
- Flexible enough to model paper planes

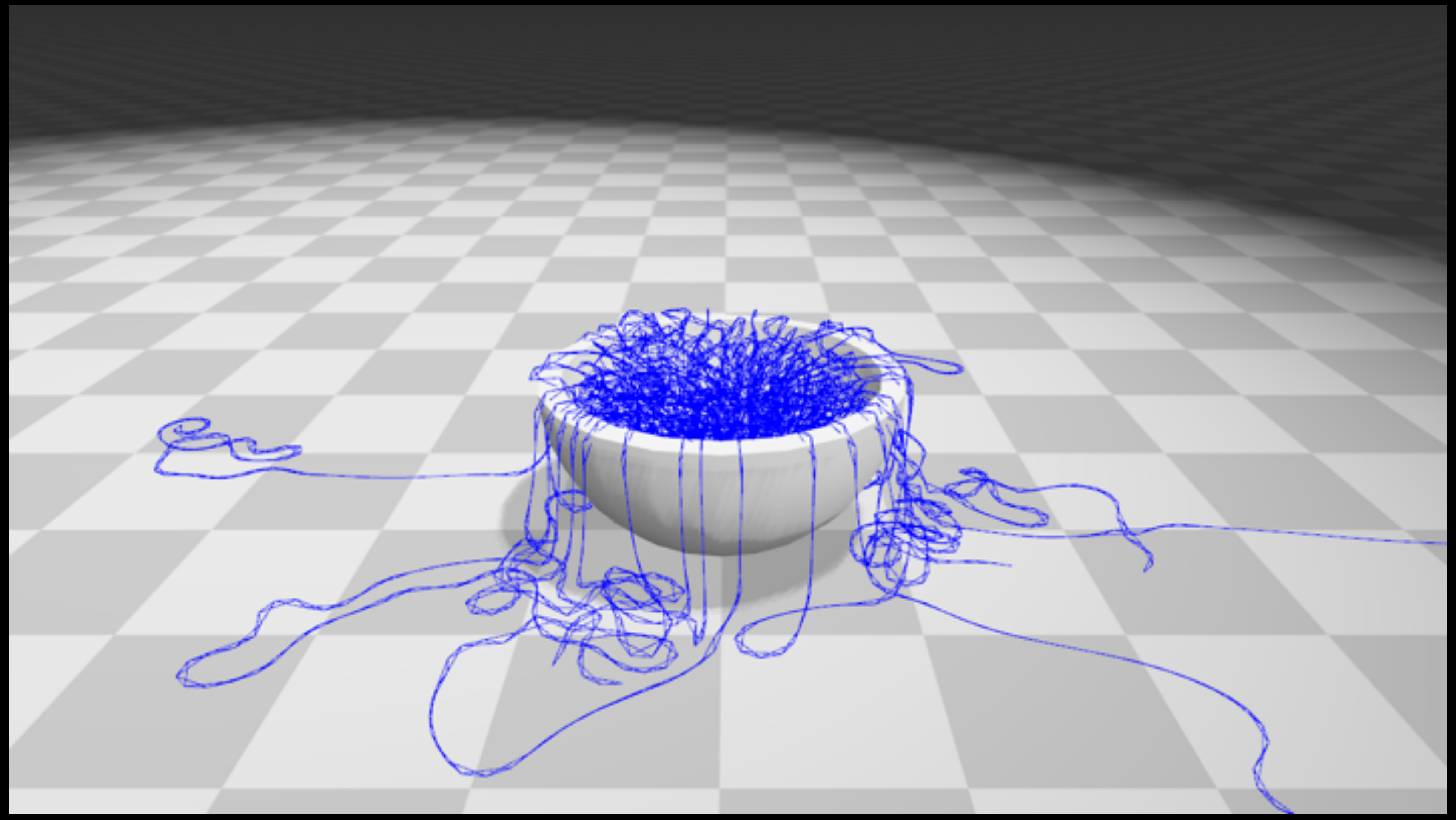


Ropes

- Build ropes from distance + bending constraints
- Fit Catmull-Rom spline to points
- Torsion possible [Umetani 14]







Examples

Limitations and Future Work

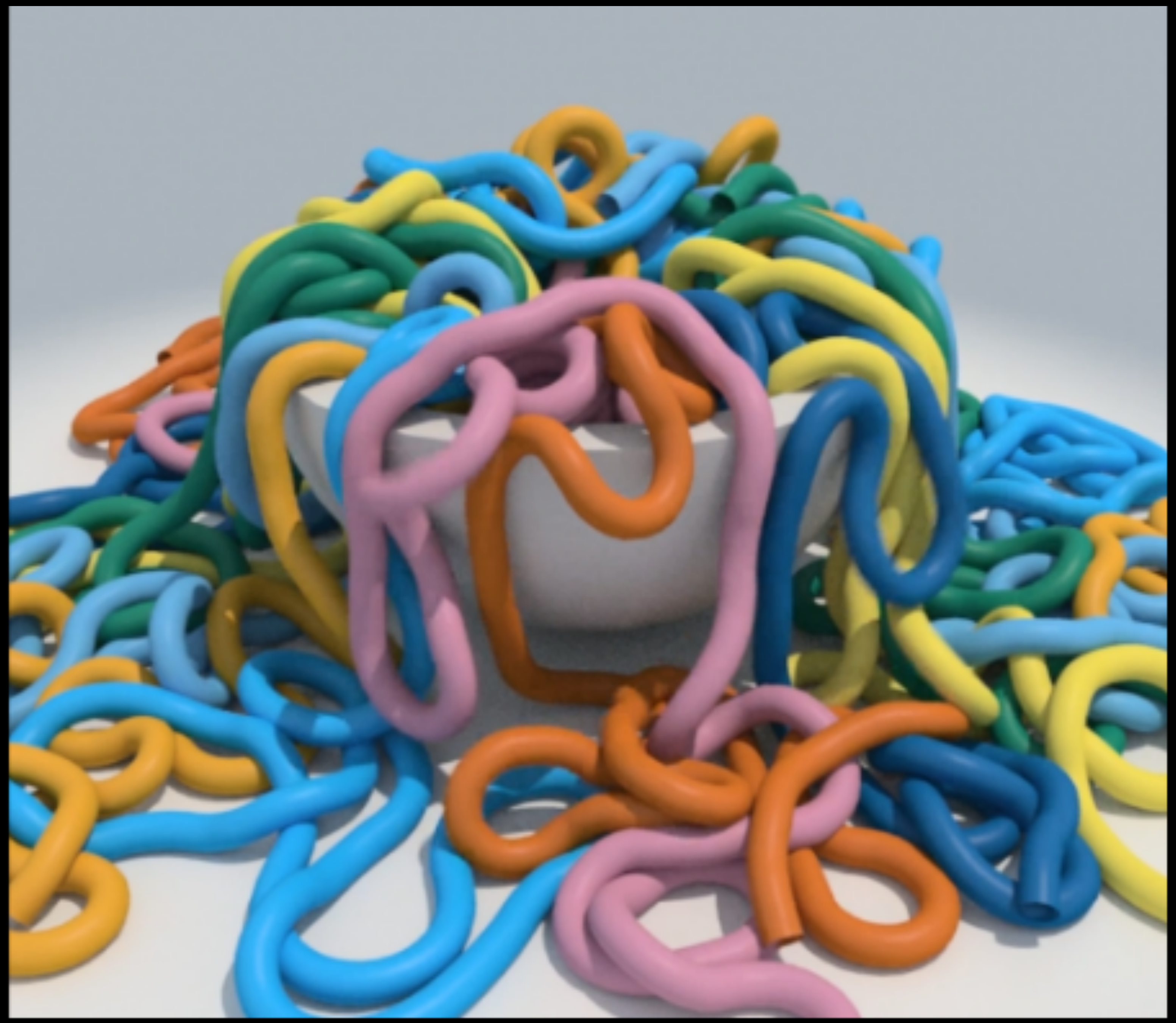
- Representing smooth surfaces problematic
- Want parallel and robust collision of simplices
- Hierarchical representation (multi-scale particles)
- Convergence for parallel solver / accelerated methods [Mazhar 2015]

Resources

- PBD available as an open source library:
  https://github.com/InteractiveComputerGraphics/PositionBasedDynamics
- Already supports many constraints: point-point, point-edge, point-triangle and edge-edge distance constraints, dihedral bending constraint, isometric bending, volume constraint, shape matching, FEM-based PBD (2D & 3D), strain-based dynamics (2D & 3D).
- Simple interface: just one class with static methods.
- MIT License
- Demos for usage

Conclusion

- Position-Based Methods are:
  - Fast, stable and simple to implement,
  - Provide a high level of control,
  - Can simulate deformable solids (1D, 2D, 3D), multi-body systems, fluids and granular materials,
  - Can be viewed as an approximation of implicit methods



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.
- Servin, M., Lacoursiere, C., & Melin, N. (2006, November). Interactive simulation of elastic deformable materials. In SIGRAD 2006. The Annual SIGRAD Conference; Special Theme: Computer Games (No. 019). Linköping University Electronic Press.
- Provot, Xavier. "Deformation constraints in a mass-spring model to describe rigid cloth behaviour." Graphics interface. Canadian Information Processing Society, 1995.
- Fratarcangeli, M., and F. Pellacini. "Scalable Partitioning for Parallel Position Based Dynamics." EUROGRAPHICS. Vol. 34. No. 2. 2015.
- Liu, Tiantian, et al. "Fast simulation of mass-spring systems." ACM Transactions on Graphics (TOG) 32.6 (2013): 214.
- Akinci, Nadir, Gizem Akinci, and Matthias Teschner. "Versatile surface tension and adhesion for SPH fluids." ACM Transactions on Graphics (TOG) 32.6 (2013): 182.
- Ryckaert, Jean-Paul, Giovanni Ciccotti, and Herman JC Berendsen. "Numerical integration of the cartesian equations of motion of a system with constraints: molecular dynamics of n-alkanes." Journal of Computational Physics 23.3 (1977): 327-341.
- Umetani, Nobuyuki, Ryan Schmidt, and Jos Stam. "Position-based elastic rods." ACM SIGGRAPH 2014 Talks. ACM, 2014.
- Müller, M., Bender, J., Chentanez, N., & Macklin, M. (2016, October). A robust method to extract the rotational part of deformations. In Proceedings of the 9th International Conference on Motion in Games (pp. 55-60). ACM.
- Bender, Jan, et al. "Position-based simulation of continuous materials." Computers & Graphics 44 (2014): 1-10.
- Unified Simulation of Rigid and Flexible Bodies Using Position Based Dynamics - VRIPHYS 2017