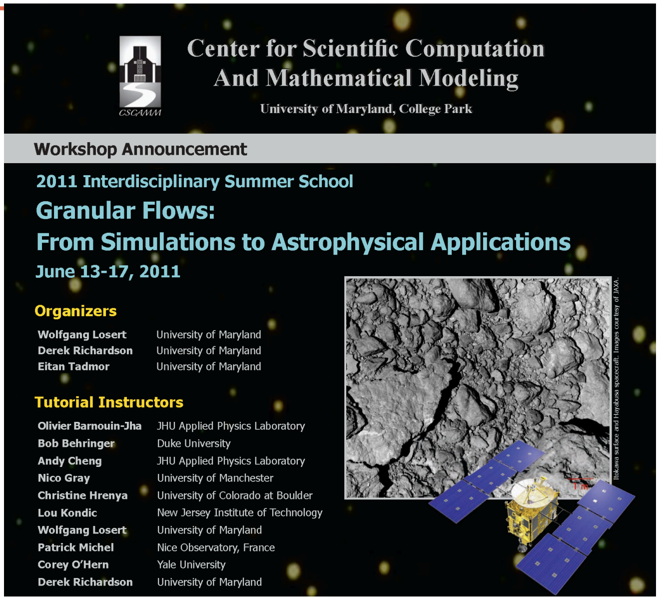
ASTEROID SIMULATIONS

Derek C. Richardson
University of Maryland



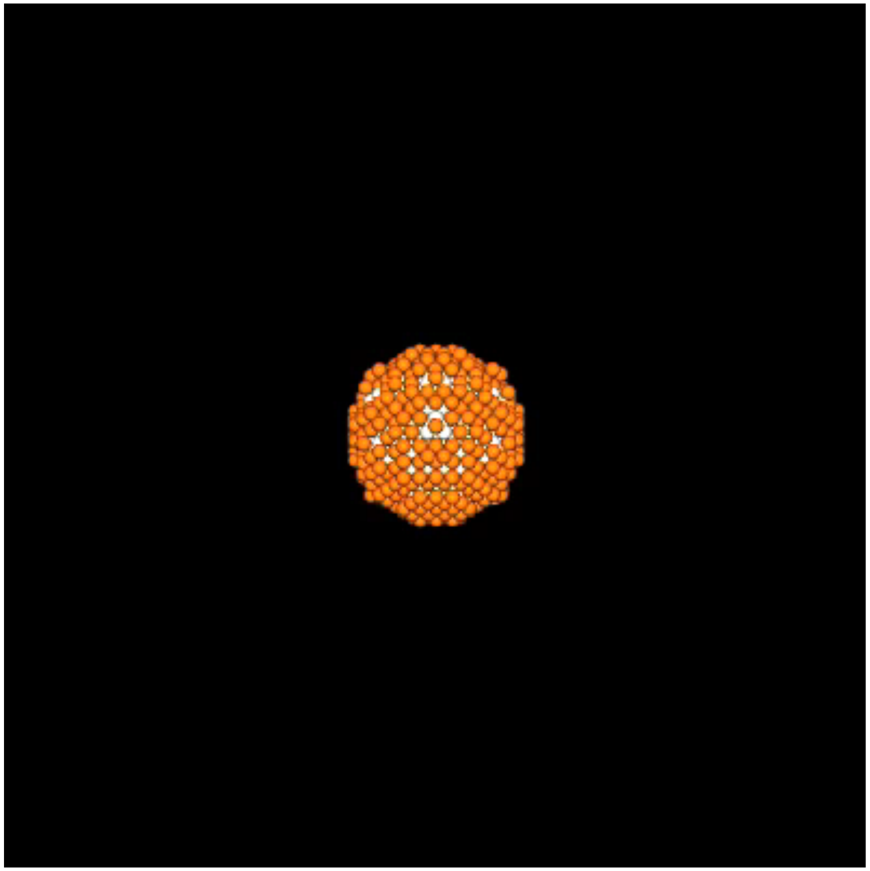
Overview

- Main topic: simulating asteroid dynamics with event-driven/hard-sphere discrete element methods.
  - Introduction to PKDGRAV.
  - Implementation details.
  - Collision resolution.
  - Complications.
  - Rigid body dynamics.

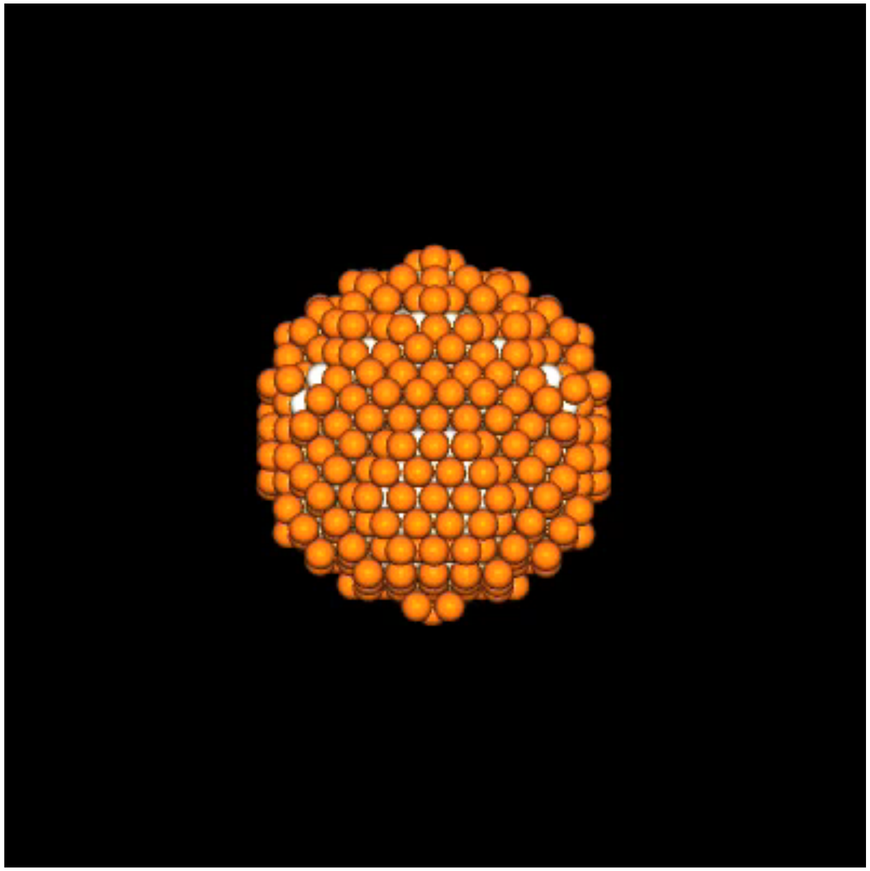
Walsh et al. 2011, Adv. Sci. Lett. 4, 311.
http://www.astro.umd.edu/~dcr/reprints.html

Example: Binary Asteroid Formation

Top View

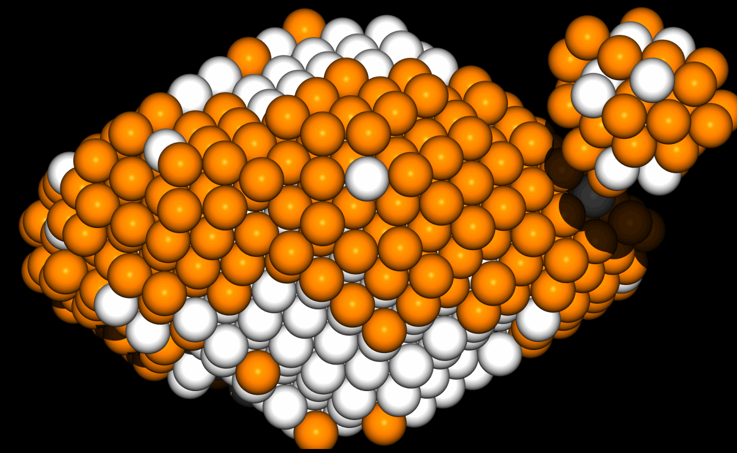


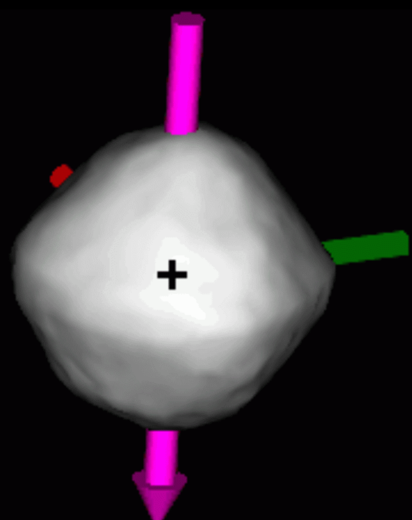
Side View



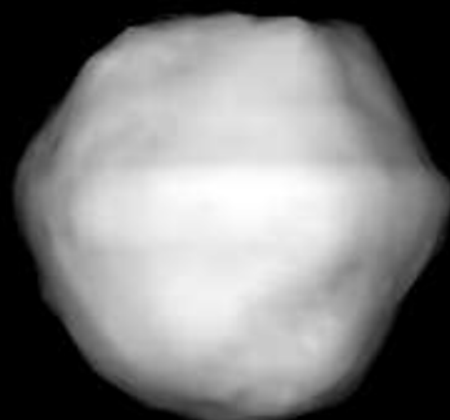
1999 KW4 Radar model, Ostro et al. 2005

YORP Spinup sims, Walsh et al. 2008





Single Asteroid RQ36
Howell et al. 2008, ACM



Binary 2004 DC
Taylor et al. 2008, ACM

Šteins from Rosetta Images

Introduction to PKDGRAV

- "Parallel *k*-D tree GRAVity code"
  - Combines parallelism and a tree code to compute interparticle forces rapidly.
- PKDGRAV solves the equations of motion for gravity (point masses):

$$\ddot{\mathbf{r}}_i = -\sum_{j \neq i} \frac{G m_j (\mathbf{r}_i - \mathbf{r}_j)}{\left|\mathbf{r}_i - \mathbf{r}_j\right|^3}$$

m = mass
r = vector position

- Started as pure cosmology code written at U Washington —not freely available. ☹

PKDGRAV Integrator

- Second-order leapfrog scheme (particle *i*, step *n*):

$$\dot{\mathbf{r}}_{i,n+1/2} = \dot{\mathbf{r}}_{i,n} + (h/2)\ddot{\mathbf{r}}_{i,n} \quad \leftarrow \text{half-step "kick"}$$

New position and velocity

$$\left\{\begin{array}{ll} \mathbf{r}_{i,n+1} = \mathbf{r}_{i,n} + h\dot{\mathbf{r}}_{i,n+1/2} & \leftarrow \text{full-step "drift"} \\ \dot{\mathbf{r}}_{i,n+1} = \dot{\mathbf{r}}_{i,n+1/2} + (h/2)\ddot{\mathbf{r}}_{i,n+1} & \leftarrow \text{half-step "kick"} \end{array}\right.$$

- Can choose timestep interval *h* based on dynamical time associated with bulk mass density *ρ* (single- or multistep):

$$h \cong \frac{0.03}{\sqrt{G\rho}}.$$

Hard-sphere Discrete Element Method (HSDEM)

- For planetesimal dynamics problems, often need to consider particle collisions.
  - The "discrete elements" are the particles themselves, sometimes representing entire bodies, or pieces of a larger body.
- Introduce hard-sphere collision condition:

Separation

$$\left|\mathbf{r}_i - \mathbf{r}_j\right| = s_i + s_j.$$

Sum of radii

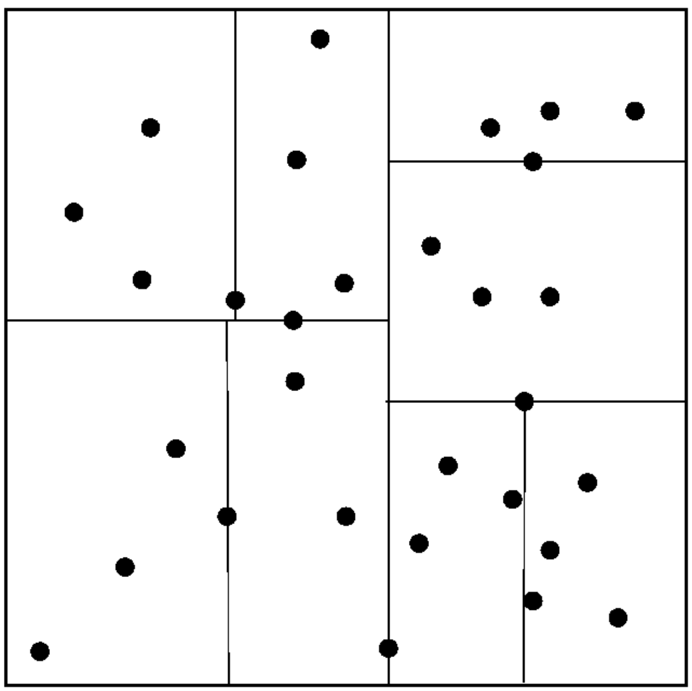
- Challenge: need to predict when collisions occur, so need efficient *neighbor-finding algorithm*.

Neighbor Finding

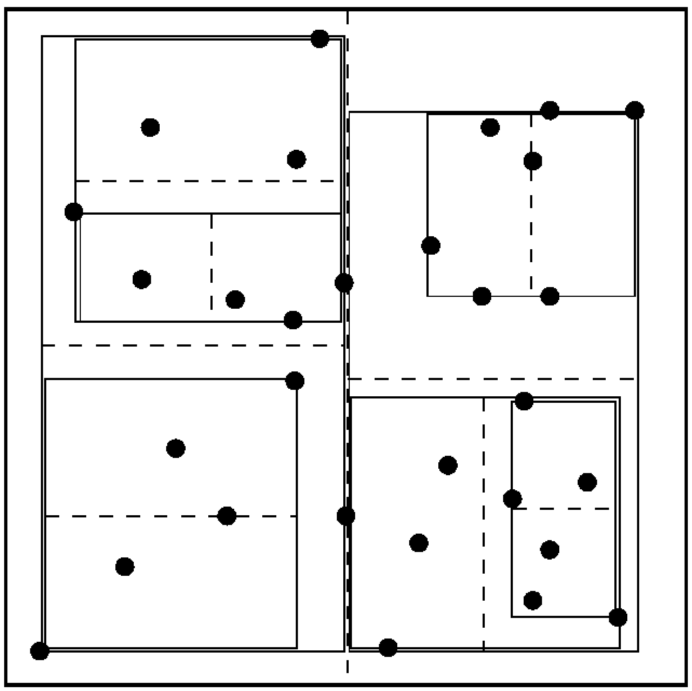
- To check all particle pairs for possible collision carries the same penalty as direct force summation: $O(N^2)$.
- Instead, use the tree code to reduce to ~$O(N \log N)$.
  - Collision search then becomes an SPH-like "smoothing" operation.
- All collisions that could occur during time h considered.
  - Collider neighbor list reset after each collision to ensure no misses.
- Perform collision search at beginning of "drift" step.
  - Exploit linear update of particle positions: line intersection.
  - Essentially simulations are event-driven *within* each timestep.

Special Section: PKDGRAV Details

Spatial Binary Tree



k-D Tree

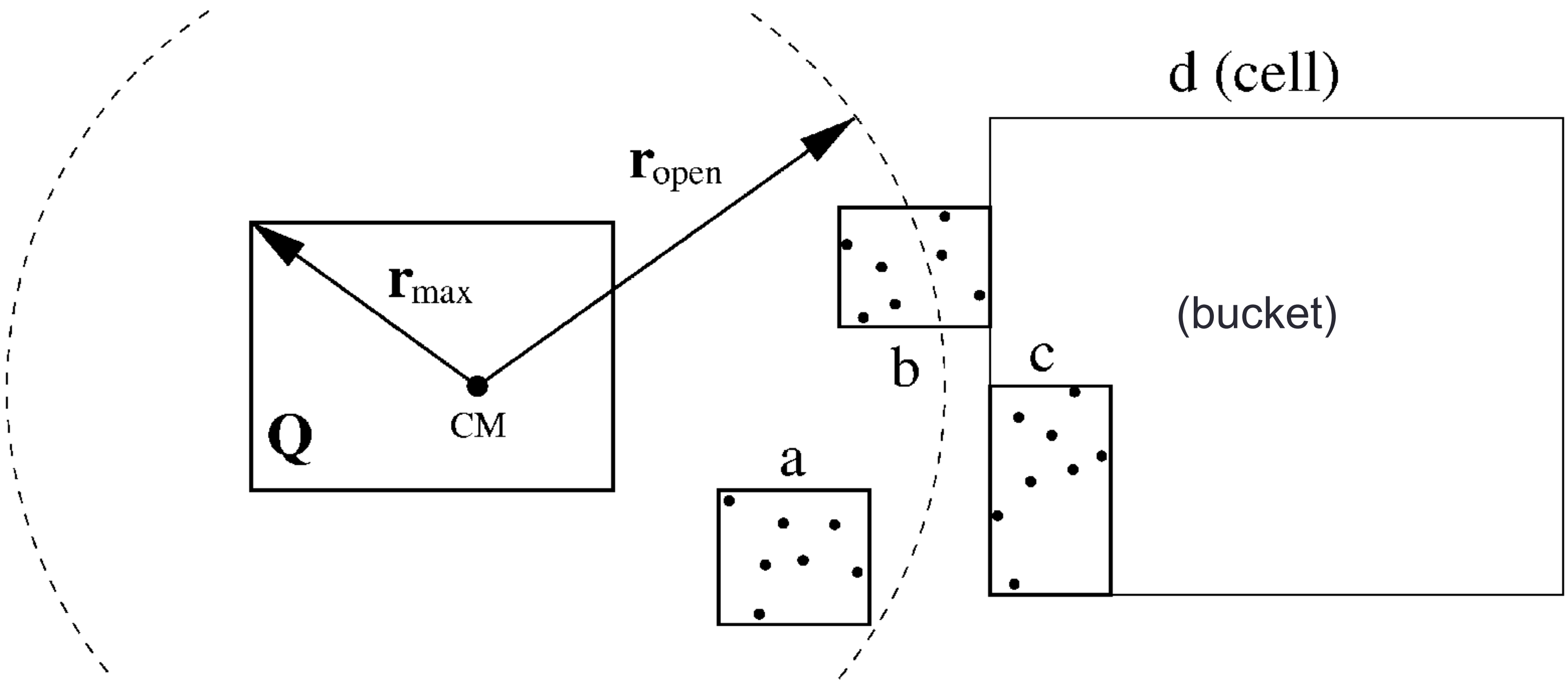


Spatial Binary Tree with Squeeze

Tree Walking

- Construct particle-particle and particle-cell interaction lists from top down for particles one bucket at a time.
- Define opening ball (based on *critical opening angle* θ) to test for cell-bucket intersection.
  - If bucket outside ball, apply multipole (c-list).
  - Otherwise open cell and test its children, etc., until leaves reached (which go on p-list).
- Nearby buckets have similar lists: amortize.

Tree Walking



Note multipole Q acceptable to all particles in cell d.

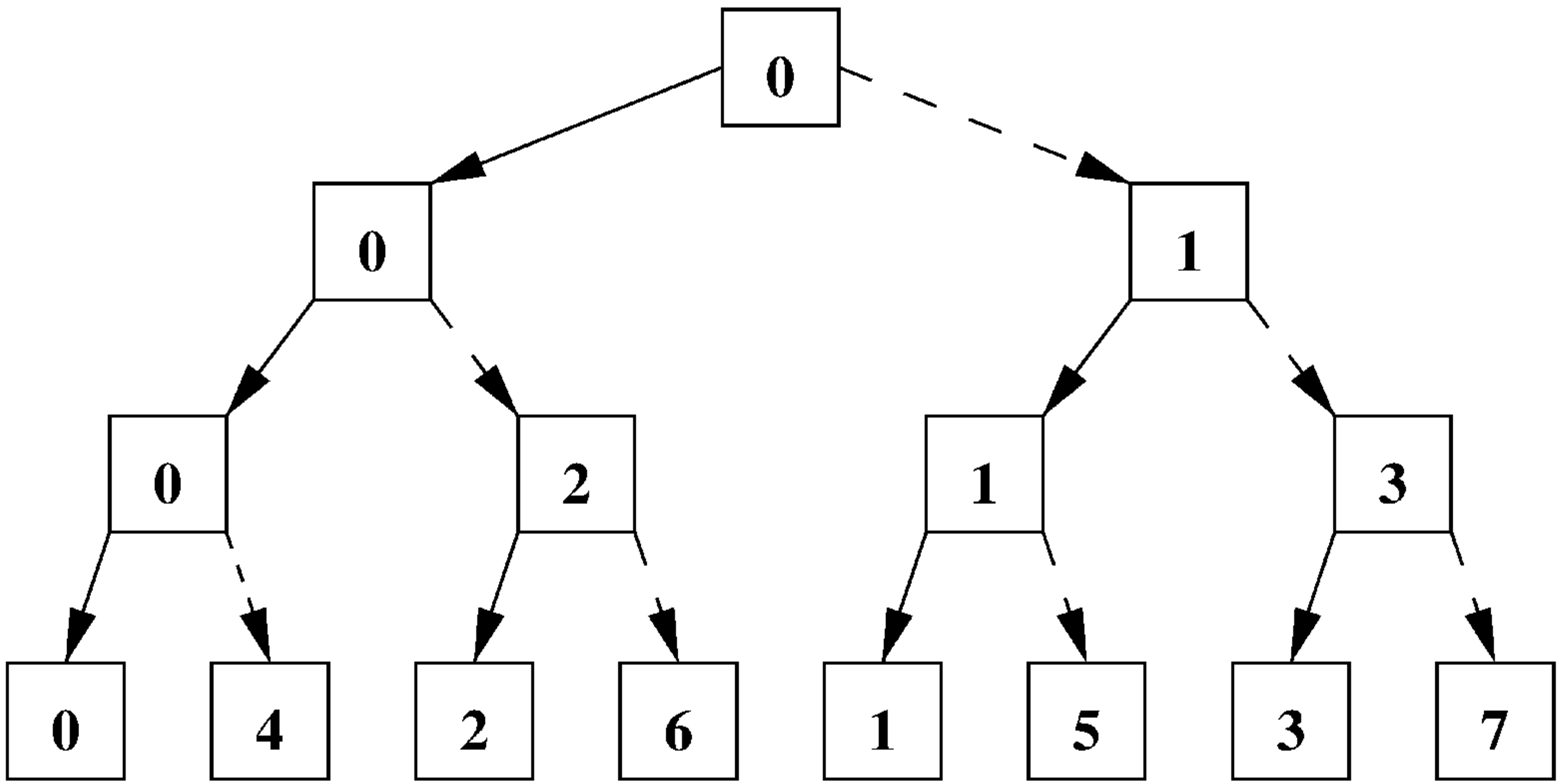
Other PKDGRAV Features

- Multipole expansion order.
  - Use hexadecapole (best bang for buck).
- Force softening (for cosmology).
  - Use spline-softened gravity kernel.
- Periodic boundary conditions.
  - Ewald summation technique available.
- Time steps.
  - Multistepping available (adaptive leapfrog).

Parallel Implementation

- Master layer (serial).
  - Controls overall flow of program.
- Processor Set Tree (PST) layer (parallel).
  - Assigns tasks to processors.
- Parallel *k*-D (PKD) layer (serial).
  - MIMD execution of tasks on each processor.
- Machine-dependent Layer (MDL, separate functions).
  - Interface to parallel primitives.

Domain Decomposition



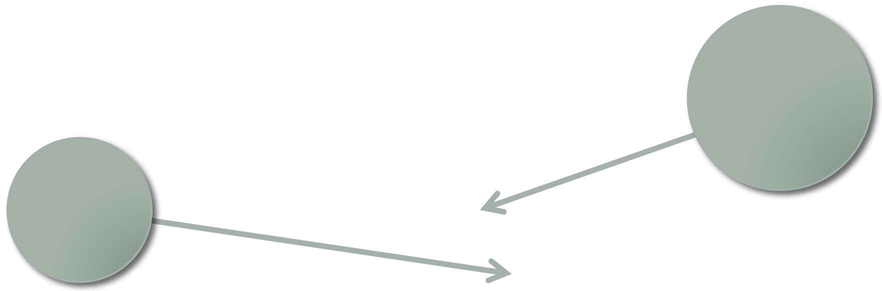
Binary tree balanced by work factors. Nodes construct local trees.

End of Special Section

Back to collisions…

- How many neighbors to search?
  - Close-packed equal-size spheres have a maximum of 12 touching neighbors.
  - For less-packed situations, only concern is a more distant fast-moving particle.
  - Typically use $N_s \sim 16$–32, with h small enough to ensure no surprises.
  - Can also search for all neighbors within a ball radius (e.g. $R \sim vh$), but can end up with many more neighbors to check.

Collision Prediction



$$\mathbf{r} = \mathbf{r}_2 - \mathbf{r}_1$$
$$\mathbf{v} = \mathbf{v}_2 - \mathbf{v}_1$$

- Collision condition after interval *t*:

$$v^2 t^2 + 2(\mathbf{r} \cdot \mathbf{v})t + r^2 = (s_1 + s_2)^2$$

- Solve for *t* (take smallest positive root):

$$t = \frac{-(\mathbf{r} \cdot \mathbf{v}) \pm \sqrt{(\mathbf{r} \cdot \mathbf{v})^2 - \left[r^2 - (s_1 + s_2)^2\right] v^2}}{v^2}$$

Collision Resolution

- Post-collision velocities and spins:

$$\mathbf{v}_1' = \mathbf{v}_1 + \frac{m_2}{M}\left[(1+\varepsilon_n)\mathbf{u}_n + \beta(1-\varepsilon_t)\mathbf{u}_t\right]$$

$$\mathbf{v}_2' = \mathbf{v}_2 - \frac{m_1}{M}\left[(1+\varepsilon_n)\mathbf{u}_n + \beta(1-\varepsilon_t)\mathbf{u}_t\right]$$

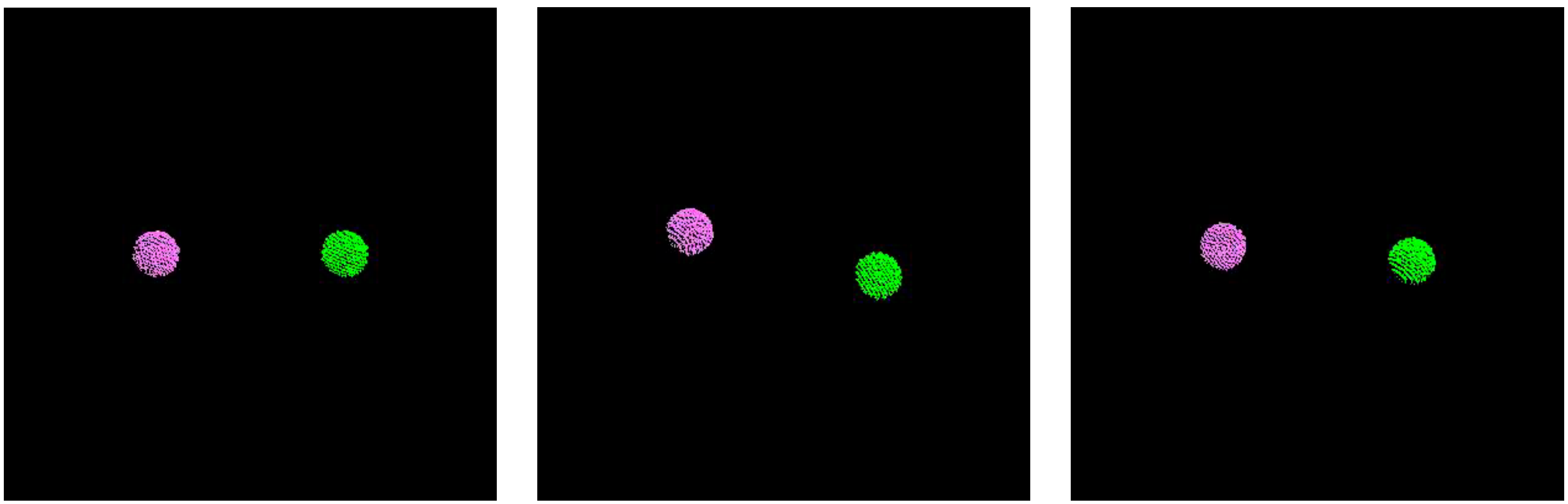
$$\boldsymbol{\omega}_1' = \boldsymbol{\omega}_1 + \beta\frac{\mu}{I_1}(1-\varepsilon_t)(\mathbf{s}_1 \times \mathbf{u})$$

$$\boldsymbol{\omega}_2' = \boldsymbol{\omega}_2 - \beta\frac{\mu}{I_2}(1-\varepsilon_t)(\mathbf{s}_2 \times \mathbf{u})$$

where $M = m_1 + m_2$, $\mu = m_1 m_2/M$, $\mathbf{u} = \mathbf{v} + \boldsymbol{\sigma}$, $\hat{\mathbf{n}} = \mathbf{r}/r$, $\mathbf{u}_n = (\mathbf{u}\bullet\hat{\mathbf{n}})\hat{\mathbf{n}}$, $\mathbf{u}_t = \mathbf{u} - \mathbf{u}_n$, $\mathbf{s}_1 = \mathbf{s}_1\hat{\mathbf{n}}$, $\mathbf{s}_2 = \text{-}\mathbf{s}_2\hat{\mathbf{n}}$, $\boldsymbol{\sigma}_i = \boldsymbol{\omega}_i \times \mathbf{s}_i$, $\boldsymbol{\sigma} = \boldsymbol{\sigma}_2 - \boldsymbol{\sigma}_1$, $\beta = 2/7$ for spheres, and $I_i = (2/5)\, m_i\, s_i^2$.

Another Example

- Gravity + collisions between dissipative particles.
- Applications include planet formation, planetary rings, asteroid family formation, etc.



Leinhardt et al. 2000, Icarus 146, 133

What about ε_n & ε_t?


GROVE

What about ε_n & ε_t?

What about ε_n & ε_t?

Durda et al. 2011, Icarus 211, 849:
$\varepsilon_n \approx 0.85$ (ε_t = ?).

Detail: Collision Handling in Parallel

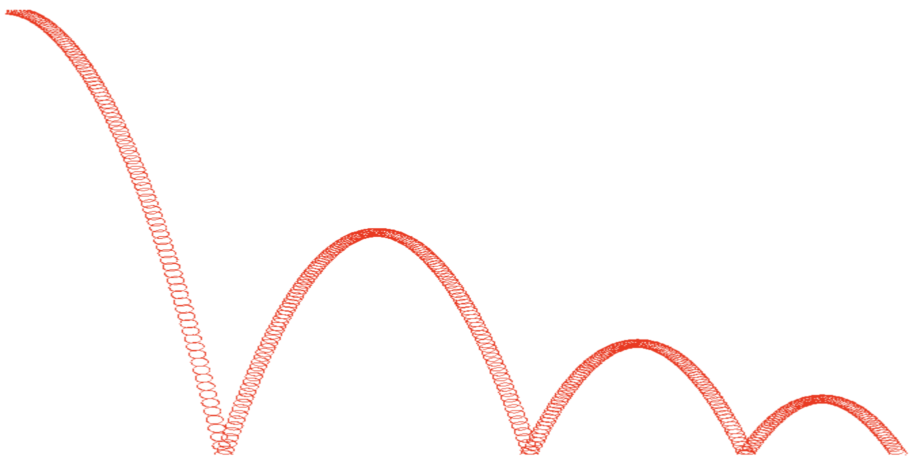
- Each processor checks its particles for next collision during current drift (could involve off-processor particle).
- Master determines which collision goes next and allows it to be carried out (possible bottleneck for dense systems!).
- Check whether any predicted collisions changed.
- Repeat until all collisions within this drift step resolved.

Complications

- The "restitution" model of billiard-ball collisions (HSDEM) is only an approximation of what really happens.
- Collisions are treated as instantaneous (no flexing) and single-point contact.
- This leads to problems:
  - Inelastic collapse.
  - Missed collisions due to round-off error.
  - No persistent contacts (friction, normal forces).

Inelastic Collapse

- A rigid ball bouncing on a rigid flat surface must come to rest, but in HSDEM this requires an infinite number of increasingly smaller bounces to occur in a finite time (Zeno's paradox!).



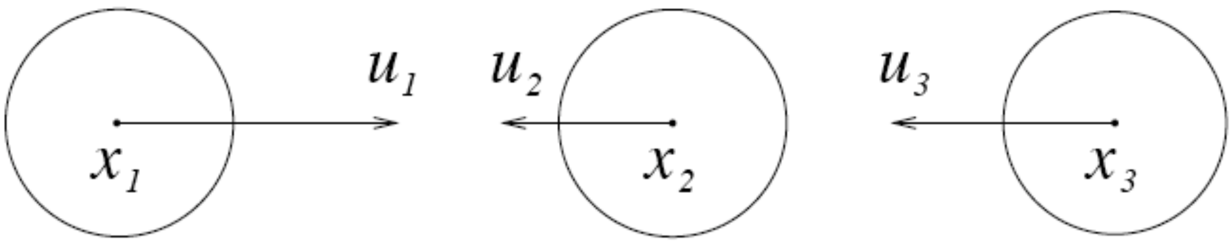
Can also occur between two self-gravitating spheres in free space…

Inelastic Collapse

- How to fix it?
  - Impose minimum speed v_{min} below which $\varepsilon_n \rightarrow 1$ (no dissipation).
  - Choose v_{min} so that this "vibration energy" is small compared to energy regimes of interest.
    - Petit & Hénon 1987a "sliding phase."

Inelastic Collapse

- Can occur in other circumstances, even *without* gravity, and at high impact speeds, e.g.



$$\begin{pmatrix} u_1'' \\ u_2'' \\ u_3'' \end{pmatrix} = \begin{pmatrix} \frac{1}{2}(1-\epsilon) & \frac{1}{2}(1+\epsilon) & 0 \\ \frac{1}{4}(1-\epsilon^2) & \frac{1}{4}(1-\epsilon)^2 & \frac{1}{2}(1+\epsilon) \\ \frac{1}{4}(1+\epsilon)^2 & \frac{1}{4}(1-\epsilon^2) & \frac{1}{2}(1-\epsilon) \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix}$$

- To collapse, the matrix must have at least one real eigenvalue between 0 & 1. This is satisfied if $0 < \varepsilon < 7 - 4\sqrt{3}$ (~ 0.072).

Inelastic Collapse, continued

- It can be shown that as $N \rightarrow \infty$, $\varepsilon_{n,\mathrm{crit}} \rightarrow 1$!
- Problem occurs in 2- & 3-D as well.
- How to fix it?
  - If distance travelled since last collision small (factor f_{crit}) compared to particle radius, set $\varepsilon_n = 1$ for next collision (e.g., $f_{\mathrm{crit}} \sim 10^{-6}$–$10^{-3}$).
  - Other strategy (not implemented): store some fraction of impact energy as internal vibration to be released stochastically.

Round-off Error and Overlaps

- Despite precautions, if there are many collisions between many particles in a timestep, round-off error can cause a collision to be missed.
- In this case, some particles may be overlapping at start of next step.
  - Minimize by good choices of h, v_{min}, and f_{crit}.
  - But sometimes that's not enough…

Round-off Error and Overlaps

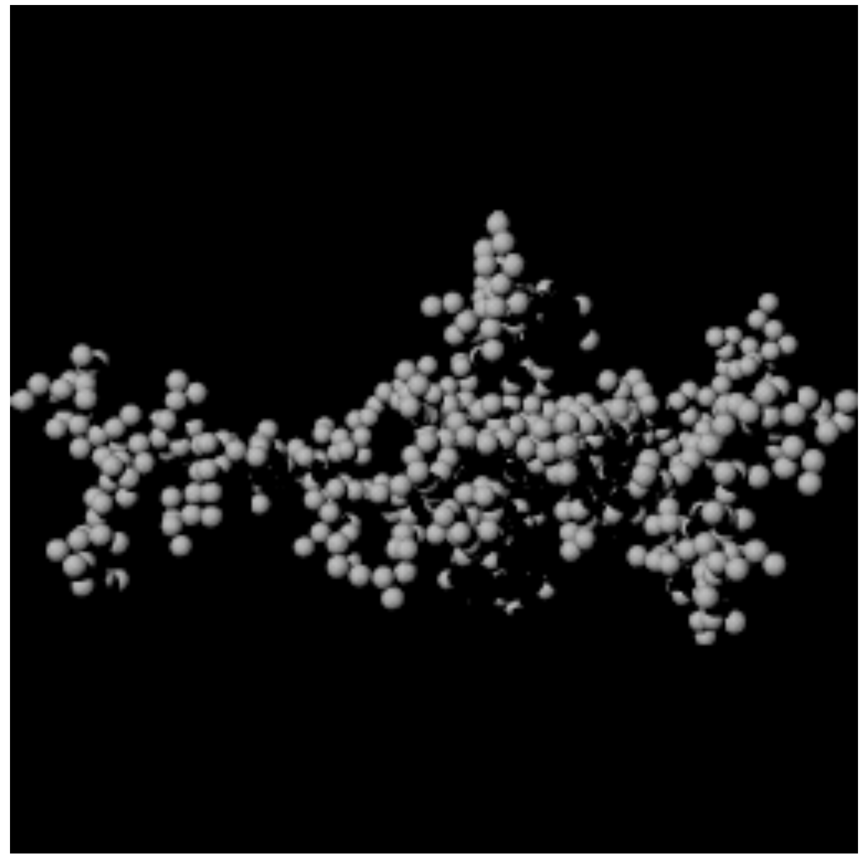
- Overlap handling strategies:
  - Abort with error (default).
  - Trace particles back in time until touching.
  - Push particles directly away until touching.
  - Merge particles (if merging enabled).
  - Apply repulsive force.
- For single particles, trace-back is best. For rigid bodies, repulsive force is best.
  - Or switch to soft-sphere DEM! But this has its own challenges… (see Thursday's lecture!).

Bonus Topic: Rigid Bodies

- Spheres are a special (easy, ideal) case.
- Perfect spheres are rarely encountered in nature, and may give misleading results when used to model granular flow, aggregation in planetary rings, etc.
- Simplest generalization: allow spheres to stick together in more complex shapes (rigid bodies). Advantages:
  - Can still use tree code for gravity & collisions.
  - Collisions are still sphere point-contact.
- Can also add breaking rules (stress response).
  - Cf. Richardson et al. 2009, P&SS 57, 183.

Rigid Bodies

- Use pseudo-particles to represent aggregate center of mass, including inertia tensor, rotation state, and orientation.
- Constituent particles constrained to move with and around center of mass —KDK only applied to pseudo-particle.
- Torques and collisions alter aggregate motion (translation + rotation).



Euler's Equations of Rigid Body Rotation

- Need to evolve spin components, according to

$$
\begin{aligned}
I_1\dot{\omega}_1 - \omega_2\omega_3(I_2 - I_3) &= N_1 \\
I_2\dot{\omega}_2 - \omega_3\omega_1(I_3 - I_1) &= N_2 \\
I_3\dot{\omega}_3 - \omega_1\omega_2(I_1 - I_2) &= N_3
\end{aligned}
$$

where I_i, ω_i are principal moments and body spin components, respectively, and N is the external torque expressed in the body frame.

Euler's Equations of Rigid Body Rotation

- Previous equations represent a set of coupled ODEs that evolve the spin axis in the body frame. Need 3 more vector equations to evolve body orientation,

$$\frac{d\hat{\mathbf{p}}_1}{dt} = \omega_3\hat{\mathbf{p}}_2 - \omega_2\hat{\mathbf{p}}_3$$

$$\frac{d\hat{\mathbf{p}}_2}{dt} = \omega_1\hat{\mathbf{p}}_3 - \omega_3\hat{\mathbf{p}}_1$$

$$\frac{d\hat{\mathbf{p}}_3}{dt} = \omega_2\hat{\mathbf{p}}_1 - \omega_1\hat{\mathbf{p}}_2$$

where $\hat{\mathbf{p}}_i$ are the principal axes of the body.

Euler’s Equations of Rigid Body Rotation

- The moments of inertia (eigenvalues) and principal axes (eigenvectors) are found by diagonalizing the inertia tensor—only need to do this when particles added to/ removed from aggregate.
- Solve this set of 12 coupled ODEs any way you like (up to next collision, or end of drift). PKDGRAV uses a fifth-order adaptive Runge-Kutta (for strongly interactive systems, dissipation not a concern).

For Completeness

- Inertia tensor:

$$\mathbf{I}_{\mathrm{agg}} = \sum_i \left[\mathbf{I}_i + m_i(\rho_i^2 - \boldsymbol{\rho}_i\boldsymbol{\rho}_i) \right],$$

with $\mathbf{I}_i = (2/5)\, m_i s_i^2 \mathbf{1}$ and $\boldsymbol{\rho}_i = \mathbf{r}_i - \mathbf{r}_a$.

- Torques:

$$\mathbf{N} = \Lambda^{\mathrm{T}} \left[\sum_{i \in a} m_i (\mathbf{r}_i - \mathbf{r}_a) \times (\ddot{\mathbf{r}}_i - \ddot{\mathbf{r}}_a) \right],$$

where the sum is over all particles in aggregate *a* and

$$\Lambda \equiv \left(\hat{\mathbf{p}}_1 \mid \hat{\mathbf{p}}_2 \mid \hat{\mathbf{p}}_3 \right).$$

Rigid Body Collisions

- Collision resolution complicated because impacts generally off-axis (non-central).
- Solutions do not permit surface friction.
  - However, off-axis collisions cause impulsive torques, allowing transfer of translational motion to rotation, and vice versa.
- Collision prediction also more complicated, due to body rotation.

Collision Prediction & Resolution

$$t = \frac{-(\mathbf{r}\bullet\mathbf{u}) \pm \sqrt{(\mathbf{r}\bullet\mathbf{u})^2 - \left[r^2 - (s_1 + s_2)^2\right]\left[u^2 + (\mathbf{r}\bullet\mathbf{q})\right]}}{u^2 + (\mathbf{r}\bullet\mathbf{q})}$$

$$\Delta\mathbf{v}_1 = \gamma(1+\varepsilon_n)(m_2 / M)\mathbf{u}_n$$

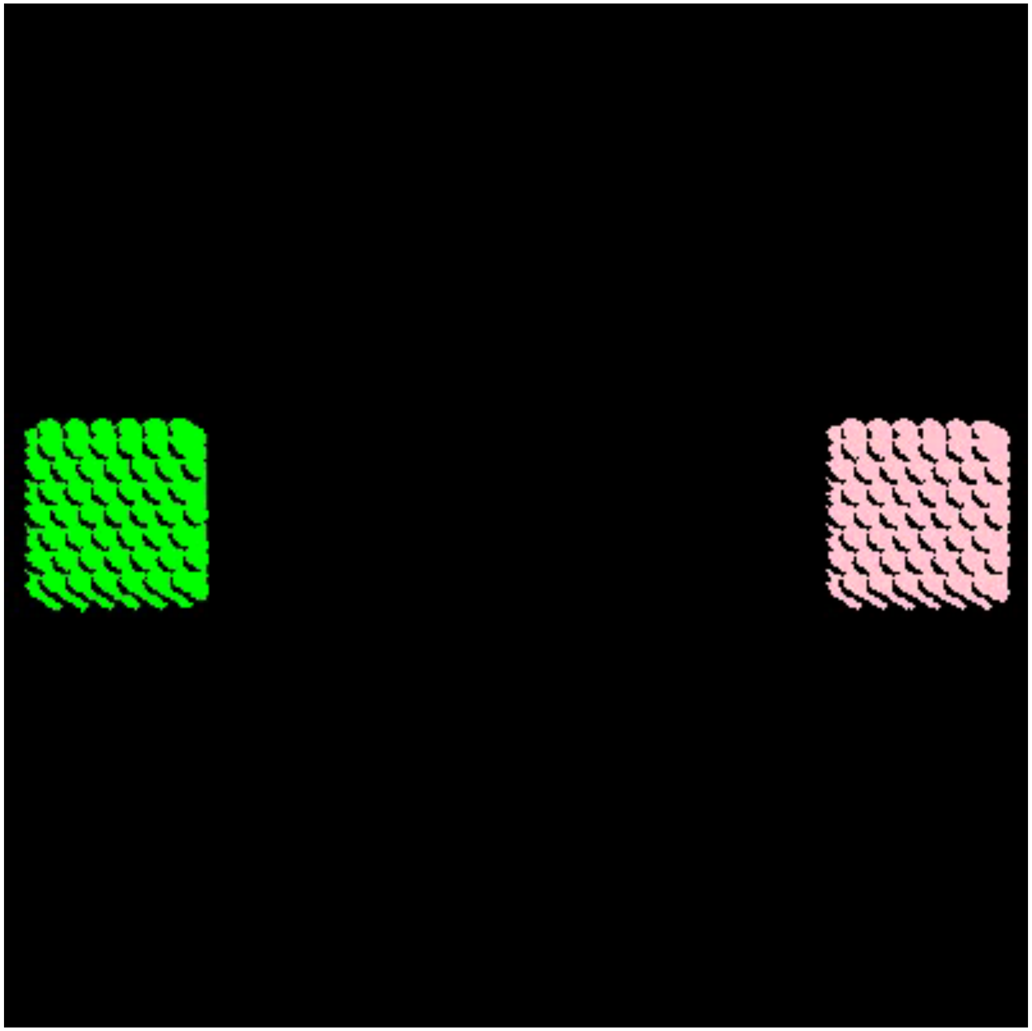
$$\Delta\mathbf{v}_2 = -\gamma(1+\varepsilon_n)(m_1 / M)\mathbf{u}_n$$

$$\Delta\boldsymbol{\omega}_1 = m_1\mathbf{I}_1^{-1}(\mathbf{c}_1 \times \Delta\mathbf{v}_1)$$

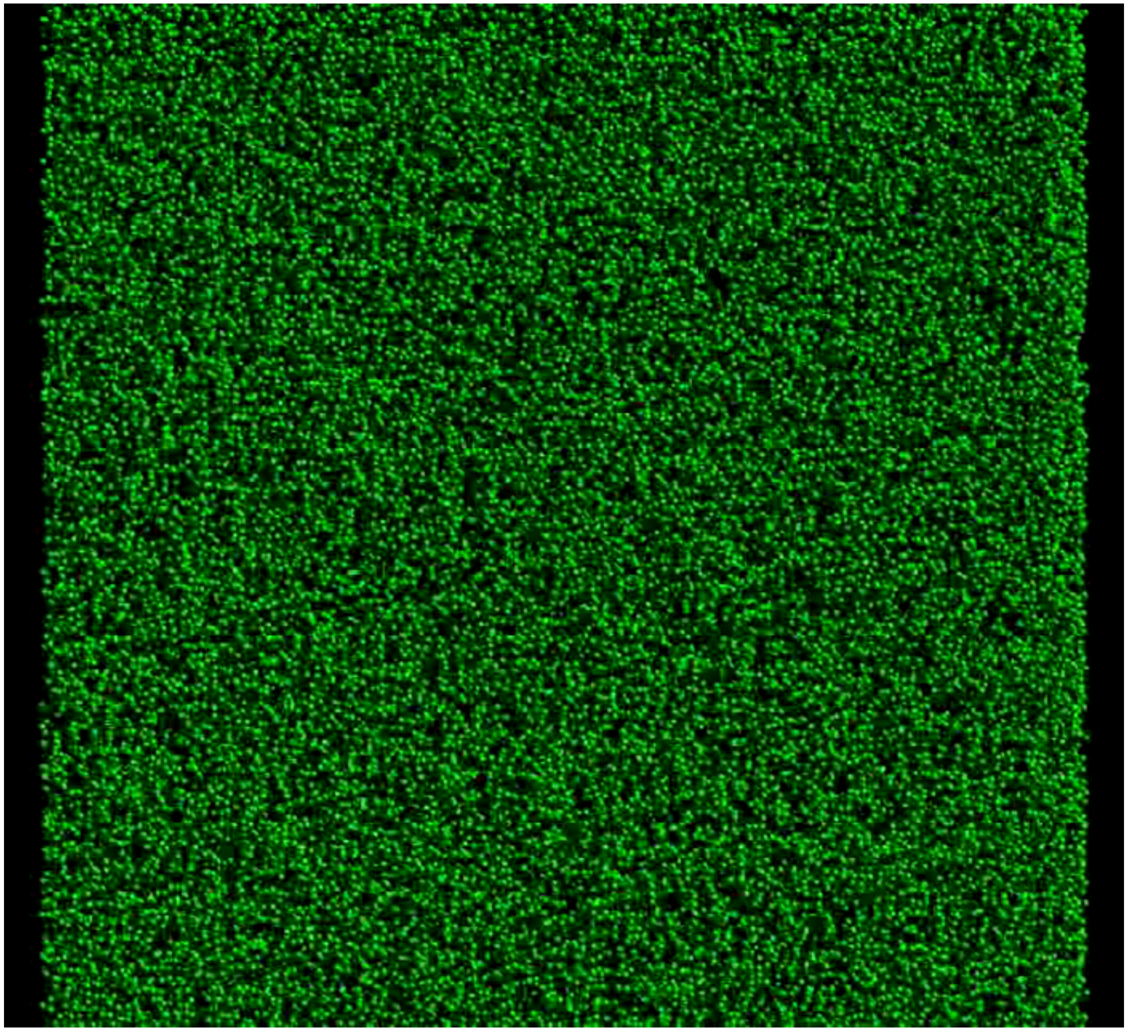
$$\Delta\boldsymbol{\omega}_2 = m_2\mathbf{I}_2^{-1}(\mathbf{c}_2 \times \Delta\mathbf{v}_2)$$

See Richardson et al. 2009 for definitions of terms!

Bouncing Cubes

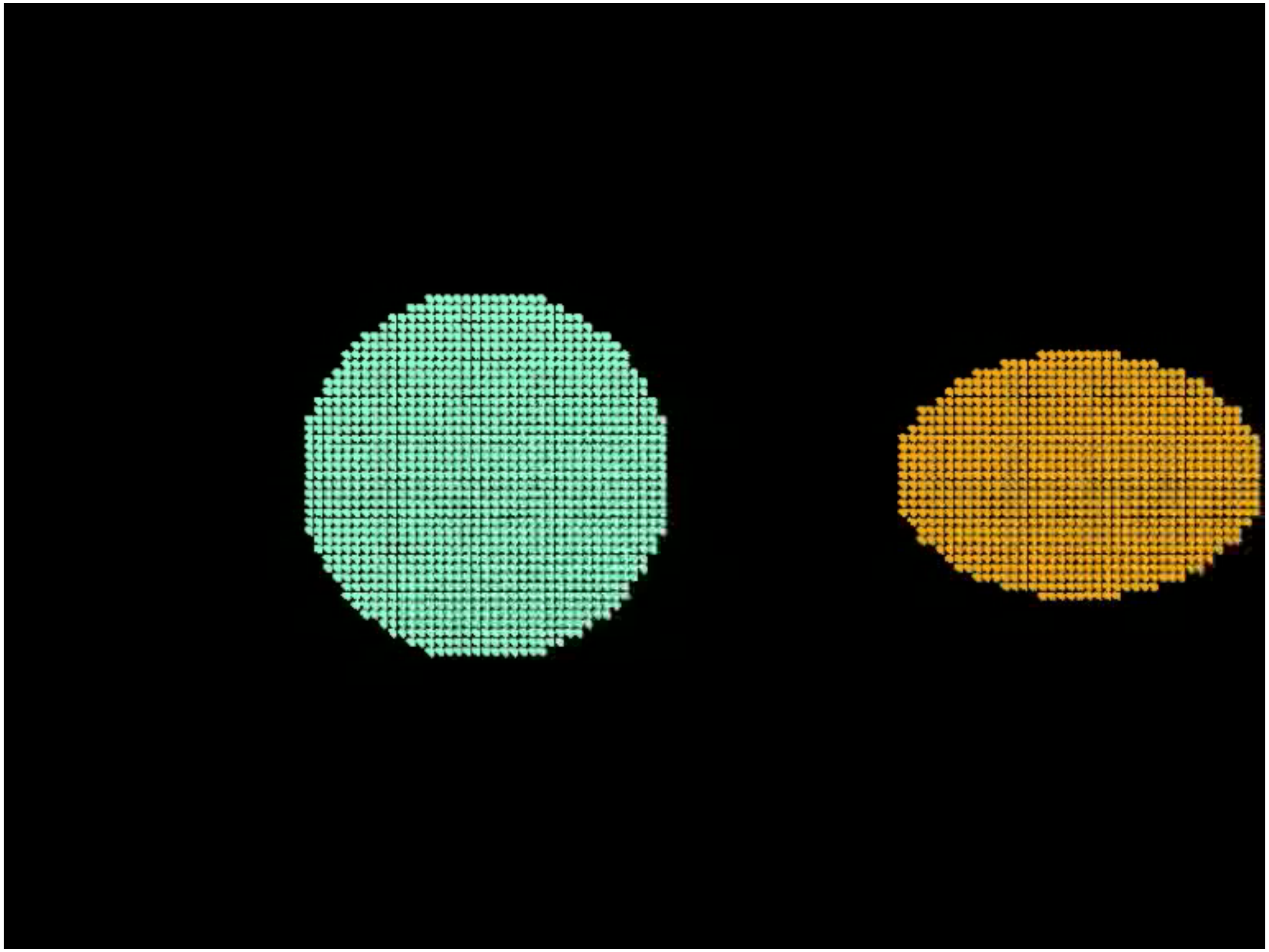


Aggregates in Rings: *Randall Perrine*

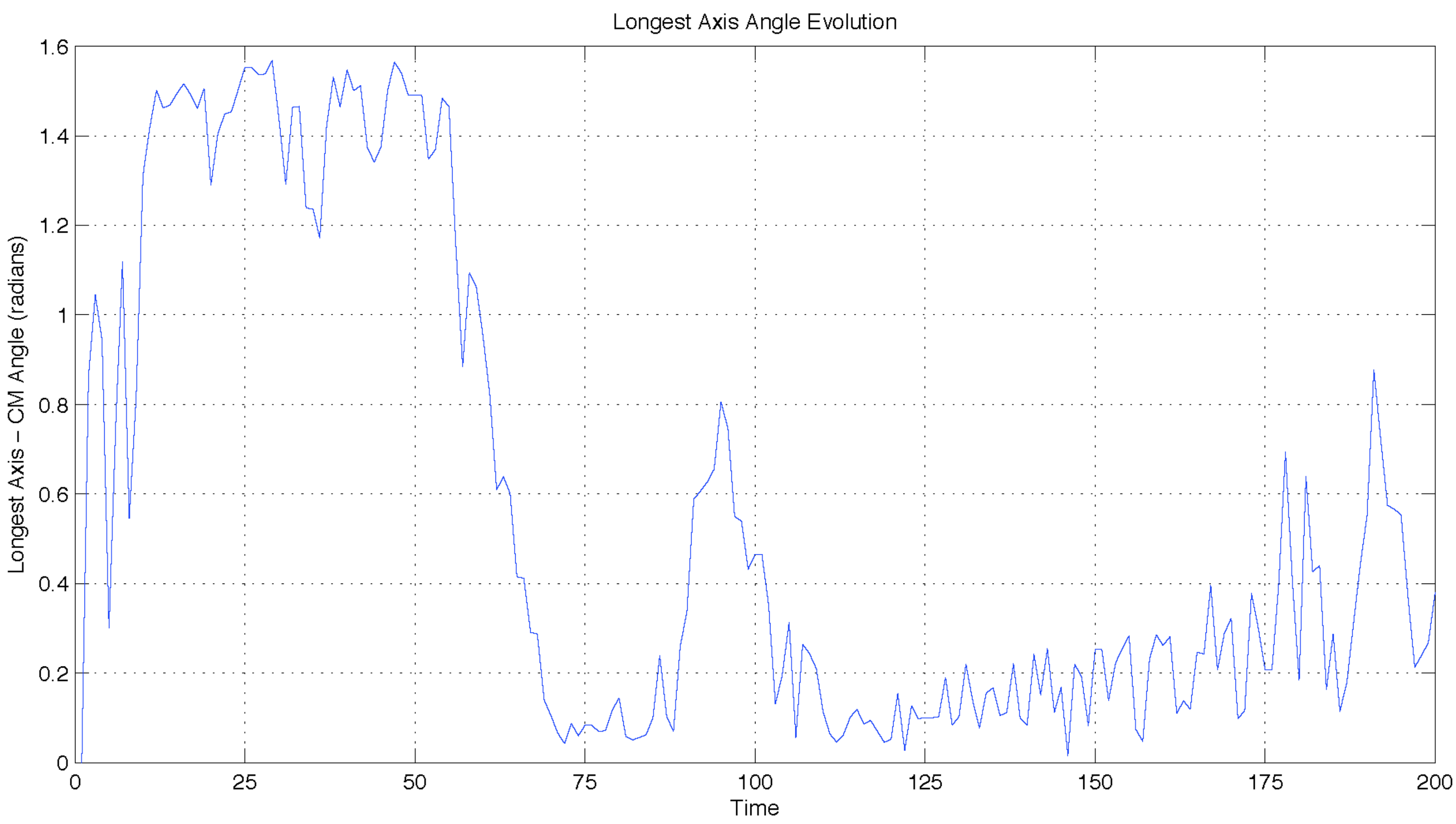


Perrine et al. 2011, Icarus 212, 719.

Testing Rigid Body Dynamics: *Brett Morris*



Testing Rigid Body Dynamics: *Brett Morris*



Summary

- HSDEM, as implemented in PKDGRAV, works well for simulations of asteroid dynamics.
  - Collisions are searched for during leapfrog drift step and carried out in time order.
  - Complications such as inelastic collapse are handled in configurable ways.
  - No explicit modeling of persistent contacts.

- Next lecture (Thursday):
  - Using HSDEM for granular mechanics: successes and failures.
  - Using soft-sphere (SS) DEM for simulations of surface processes on asteroids—implementation and preliminary results.