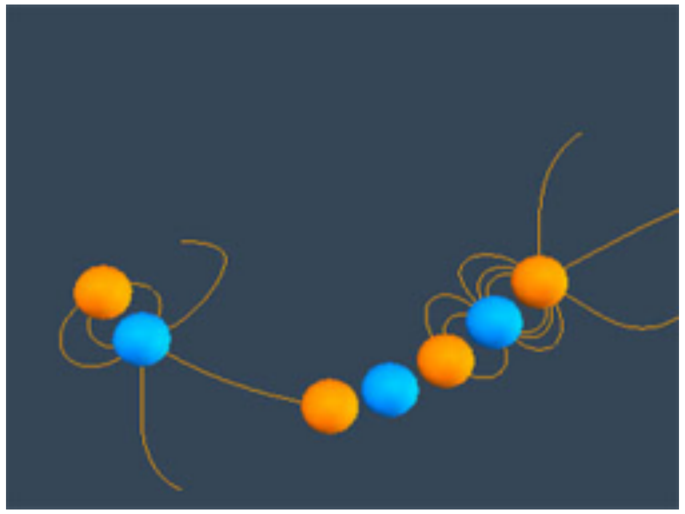
N-Body Problem

Introduction

- Many physical phenomena can be simulated with a particle system where each particle interacts with all other particles according to the laws of physics.

- Examples range from astrophysical simulations of celestial motions to electrostatic interactions of molecules.



N-Body Problem

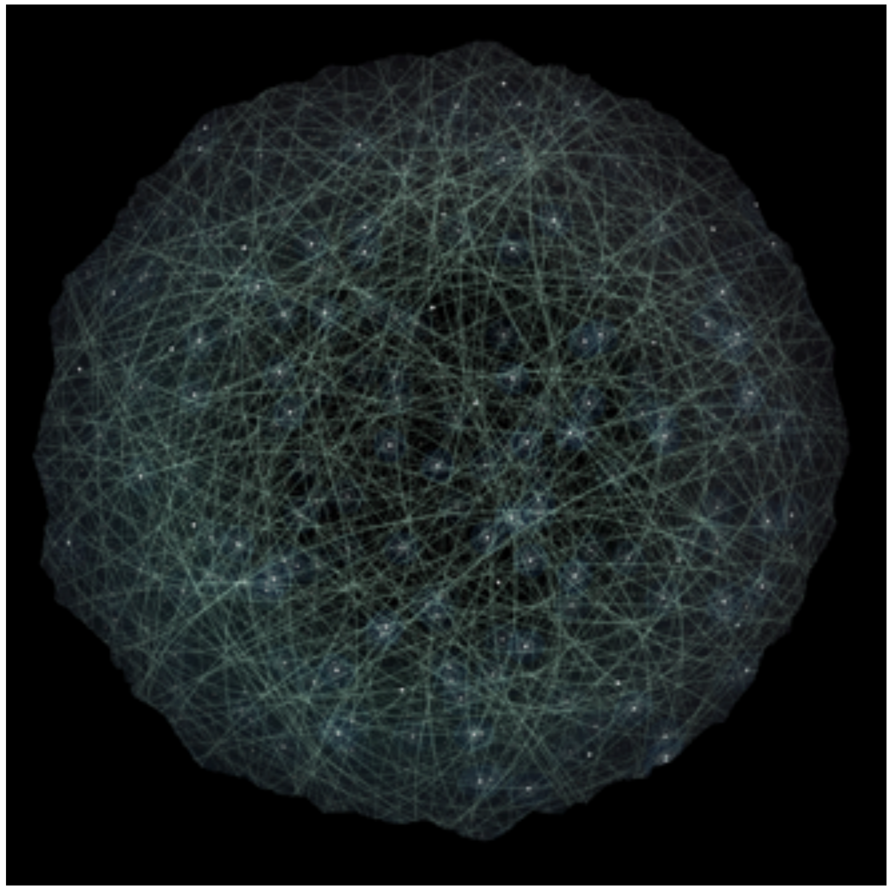
- Generally, the N-body problem is the problem of predicting the motion of a group of N objects that each independently interact with one another over a long range (usu. gravitationally or electrostatically).

- Formally, for a group of N objects in space, if the initial positions (x_0) and velocities (v_0) are known at time t_0, predict the positions (x) and velocities (v) of the N objects at a later time t.

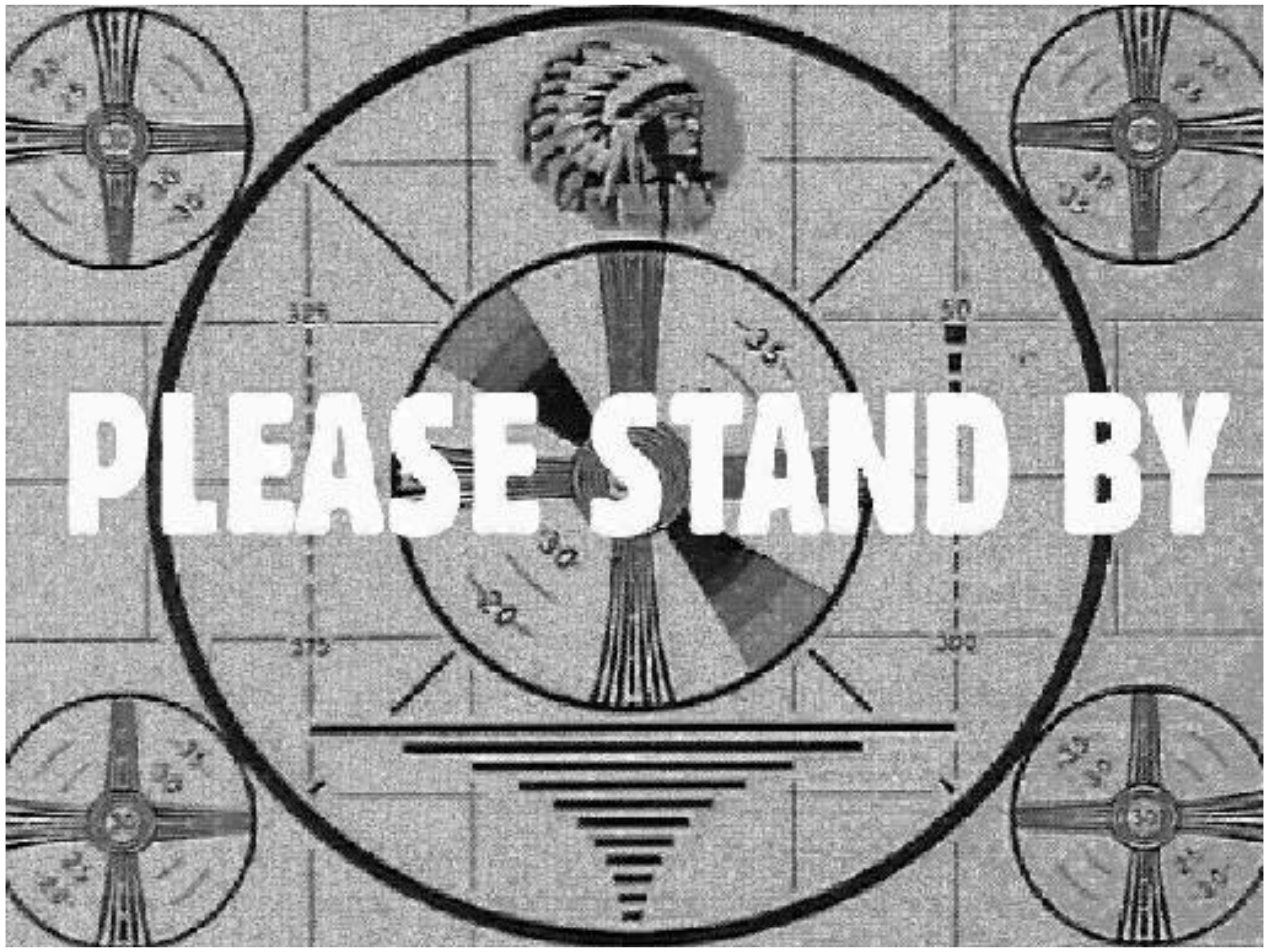
- *Prima facie*, the solution to this problem is in $O(N^2)$ because each of the N objects interacts with N-1 other objects.

- Solving this problem was originally motivated by the need to understand the motion of the Sun, planets and the visible stars, but it has been applied to galaxies, planets, fluids, and molecules (breaks down for subatomic particles).

- Its first complete mathematical formulation appeared in Isaac Newton's *Principia*.

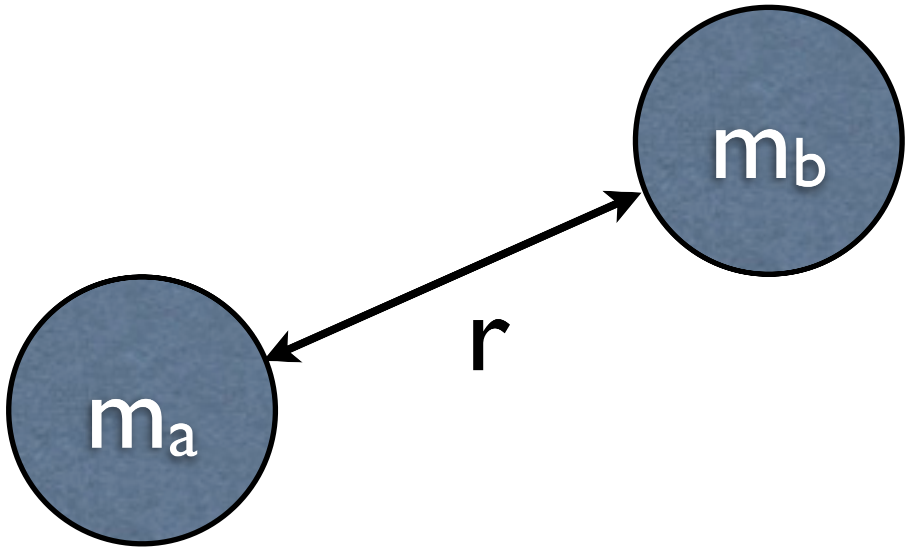


Intermission



Gravitational N-Body Problem

- Finding positions and movements of bodies in space subject to gravitational forces from other bodies using Newtonian laws of motion.



- Gravitational force F between two bodies of masses m_a and m_b is:

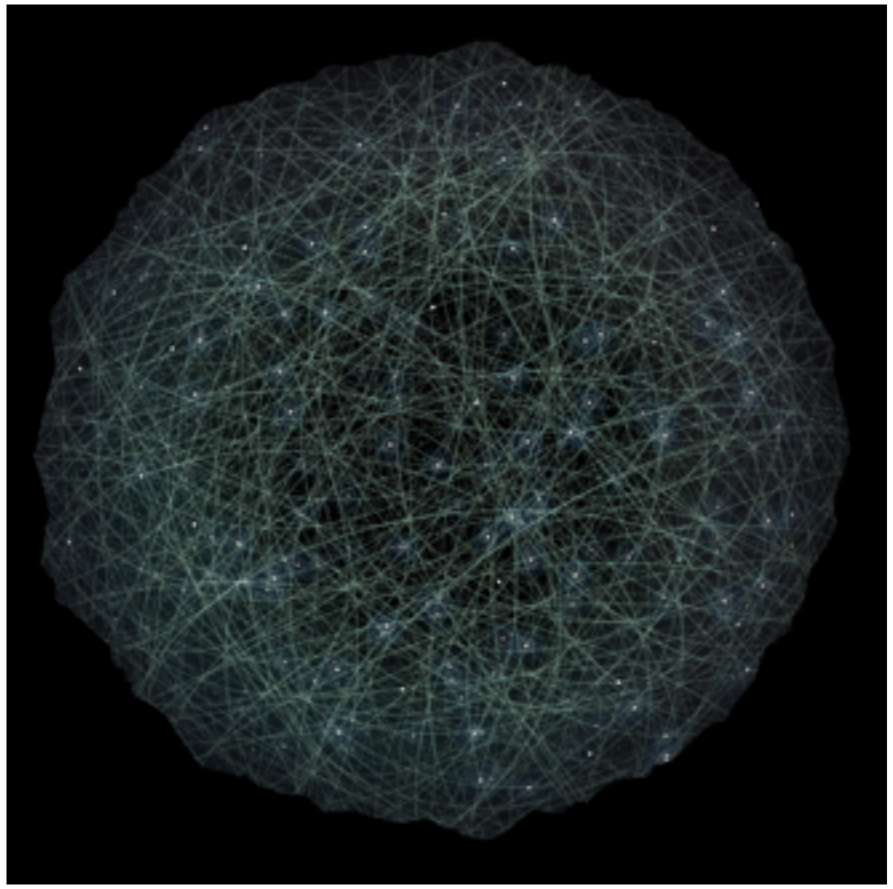
$$F_{ab} = \frac{G m_a m_b}{r^2}$$

- G is the gravitational constant (6.673 × 10^{-11} m^3 kg^{-1} s^{-2}) and r the distance between the bodies.

$$r = \sqrt{(x_b - x_a)^2 + (y_b - y_a)^2 + (z_b - z_a)^2}$$

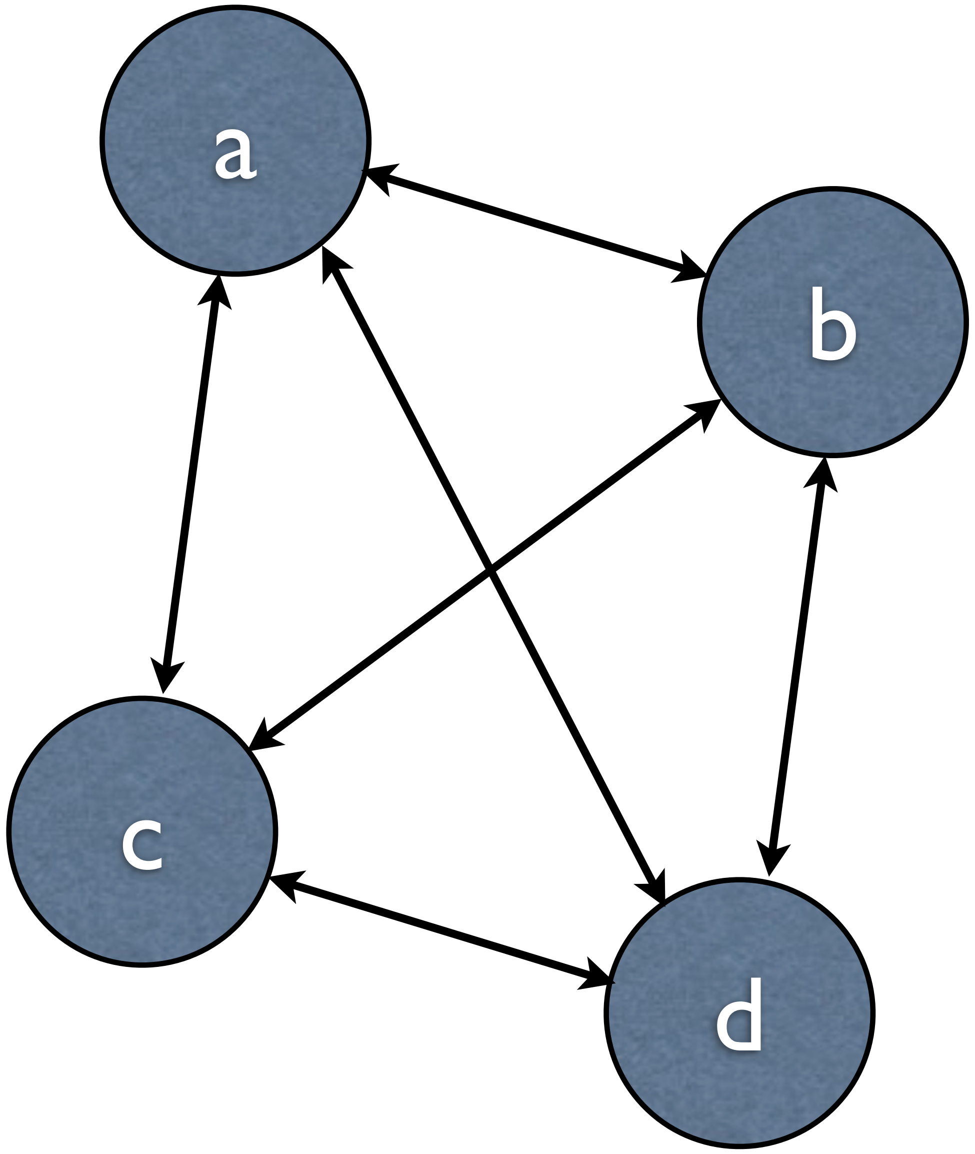
- For a system of N particles, the sum of the forces is:

$$F = \sum_{i<j} F_{ij} = \sum_{i<j} \frac{G m_i m_j}{r_{ij}^2}$$



Simple N-Body Scenario with N=4

- Thanks to Newton's Third Law ("for every action, there is an equal and opposite reaction"), the number of interactions is halved.
- However, this is still a complicated problem that only grows in complexity with N.



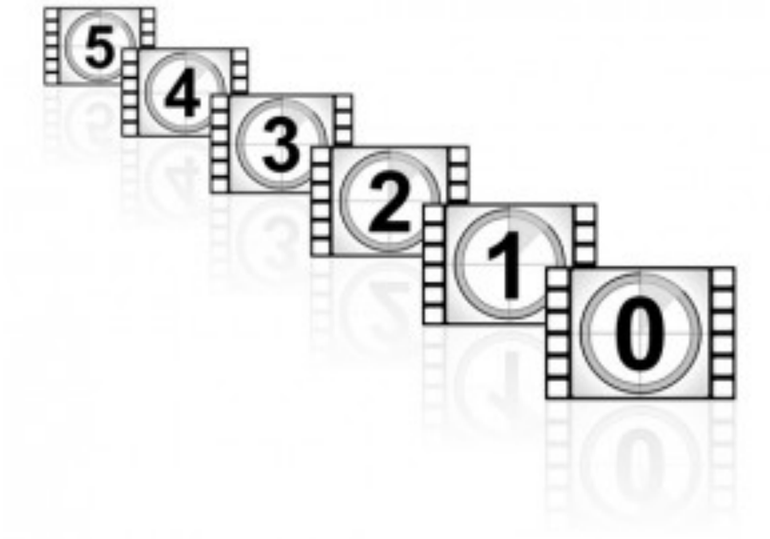
Gravitational N-Body Problem

- Subject to forces, a body accelerates according to Newton's second law: F = ma where m is mass of the body, F is force it experiences and a is the acceleration.

- Let the time interval be Δt. Let v_t be the velocity at time t. For a body of mass m the force is:

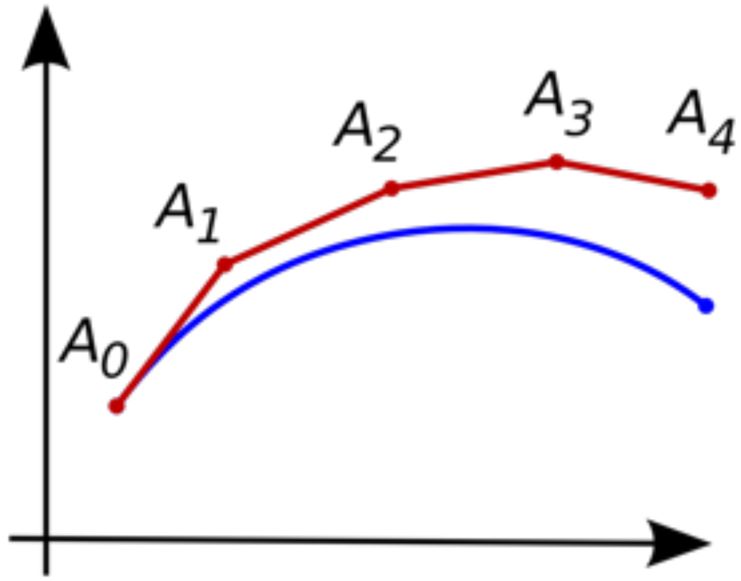
$$F = m\frac{v_{t+1} - v_t}{\Delta t}$$

Gravitational N-Body Problem



- New velocity then is

$$v_{t+1} = v_t + \frac{F \cdot \Delta t}{m}$$



- Over time interval Δt position changes by

$$x_{t+1} = x_t + v \cdot \Delta t$$

  - where x_t is its position at time t.

- Once bodies move to new positions, forces change and computation has to be repeated.

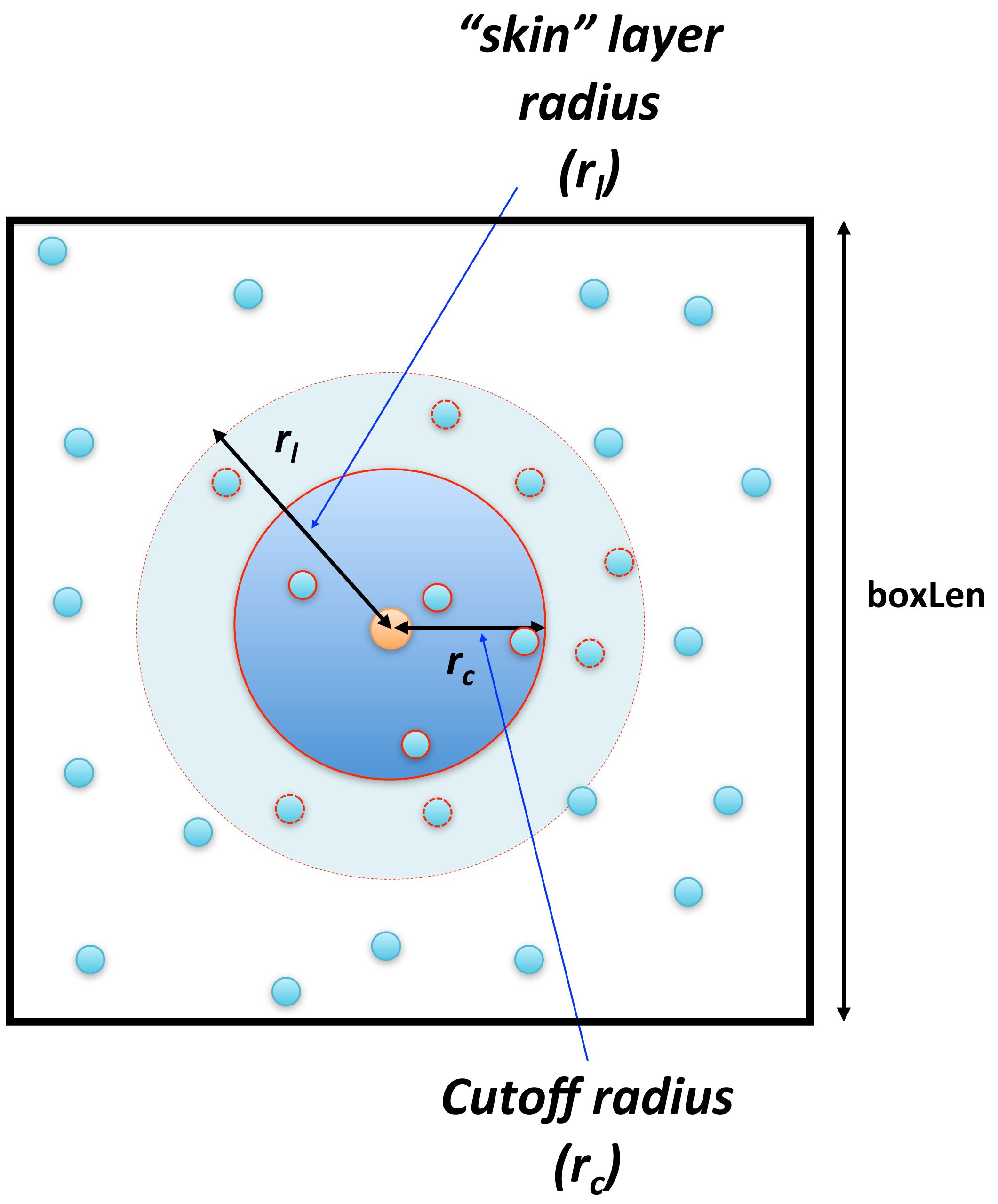
N-Body Simulation

- Overall gravitational N-body computation can be described as:

```
for (t = 0; t < tmax; t++) {                /*  time periods */
   for (i = 0; i < N; i++) {                /* for each body */
      F = Force_routine(i);                 /* force on body i from n-1
                                               other bodies (O(n)) */
      v_new[i] = v[i] + F * dt / m;         /* new velocity */
      x_new[i] = x[i] + v_new[i] * dt;      /* new position */
   }
   for (i = 0; i < N; i++) {                /* for each body */
      x[i] = x_new[i];                      /* update velocity */
      v[i] = v_new[i];                      /* and position */
   }
}
```

Verlet Neighbor List

- Make “neighbor list” of bead pairs with $r_{ij} < r_l$.
- Only compute forces for bead pairs from neighbor list with $r_{ij} < r_c$.
- Update neighbor list every N timesteps.
- r_c and r_l chosen as 2.5 σ and 3.2 σ, respectively.
- Speedup from calculating forces for only neighbor list bead pairs with $r_{ij} < r_c$.



(Verlet, 1967; Allen and Tildesley, 1987)

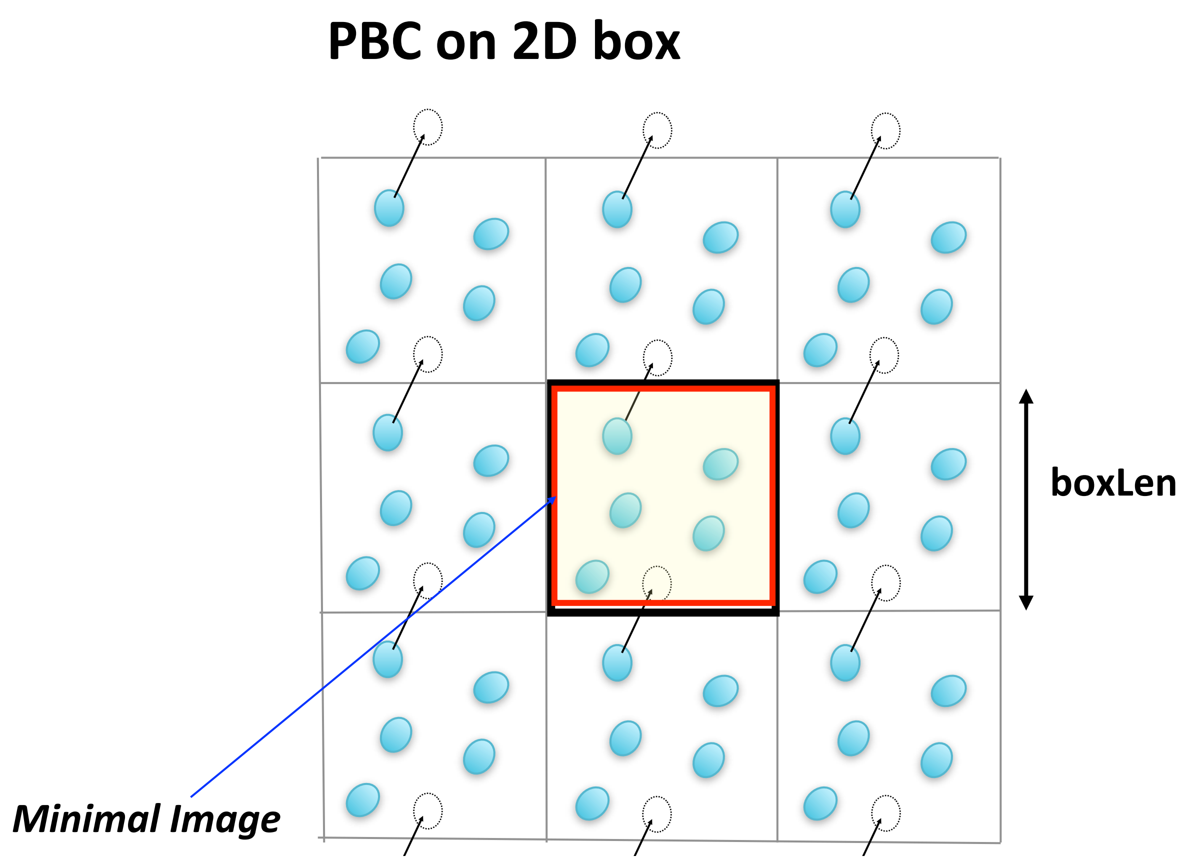
Periodic Boundary Conditions

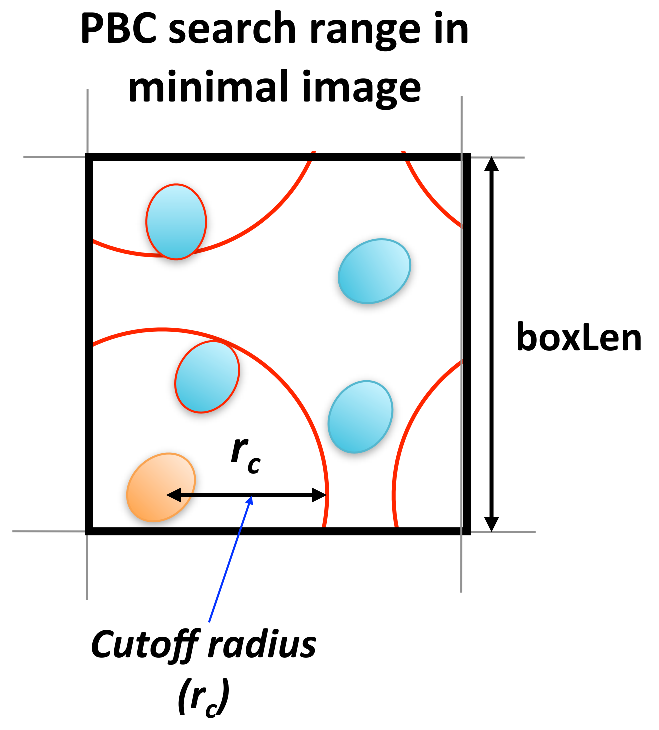
- All interactions wrap around central "image" that is repeated in all directions.

- "infinite" simulation space

- Cutoffs also wrap
  - r_c must be less than boxLen

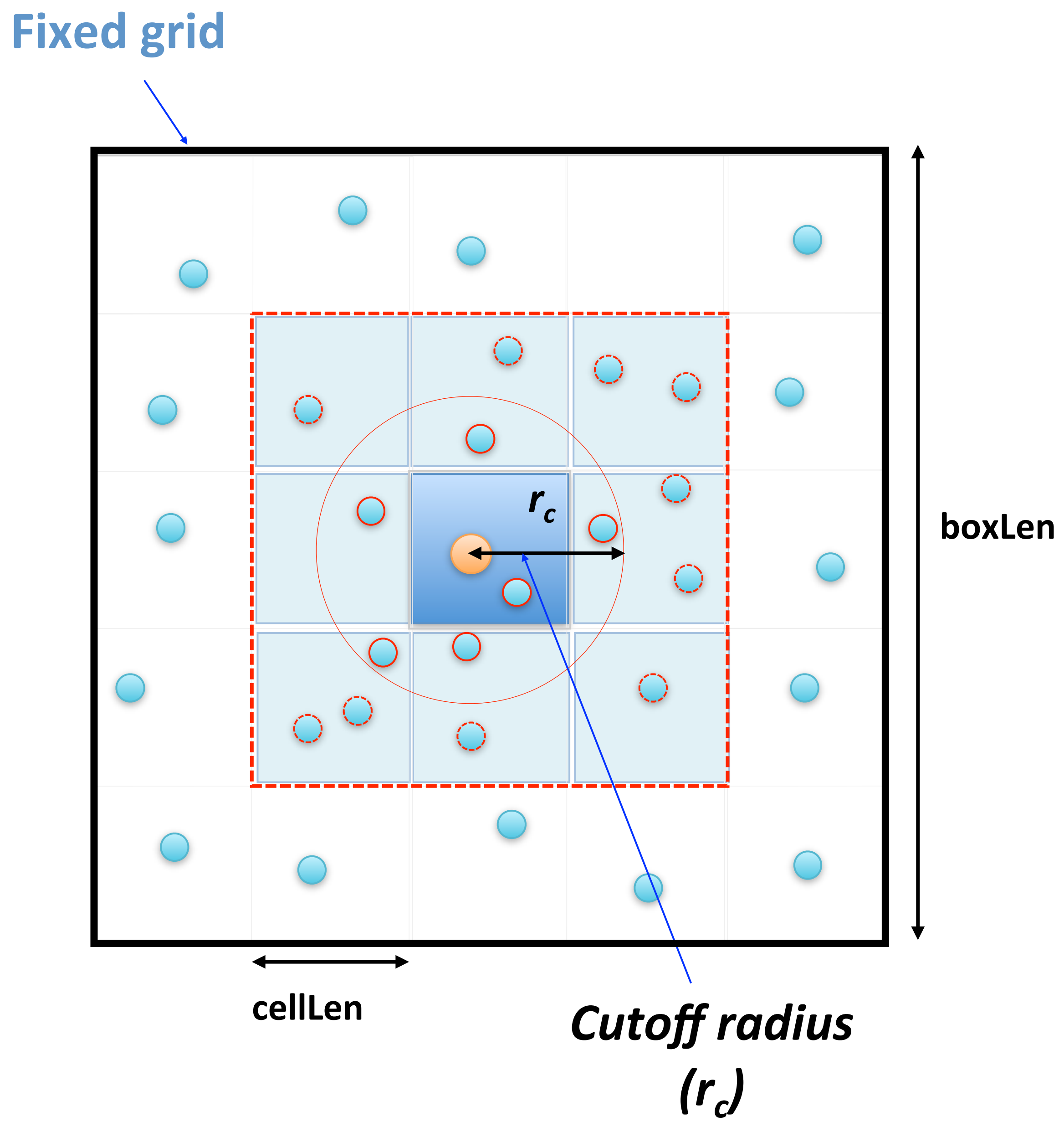
(Allen and Tildesley, 1987)





Cell List

- Split simulation box into nCell identical cells, each with length cellLen.
- Make “cell list” of bead pairs within center + neighboring cells
- Neighboring cells wrap (PBC)
- Only compute forces for bead pairs from neighbor list with $r_{ij} < r_c$.
- Update neighbor list every N timesteps.
- r_c chosen as 2.5 σ, cellLen as ½ r_c.
- Speedup from calculating forces for only cell list bead pairs with $r_{ij} < r_c$.

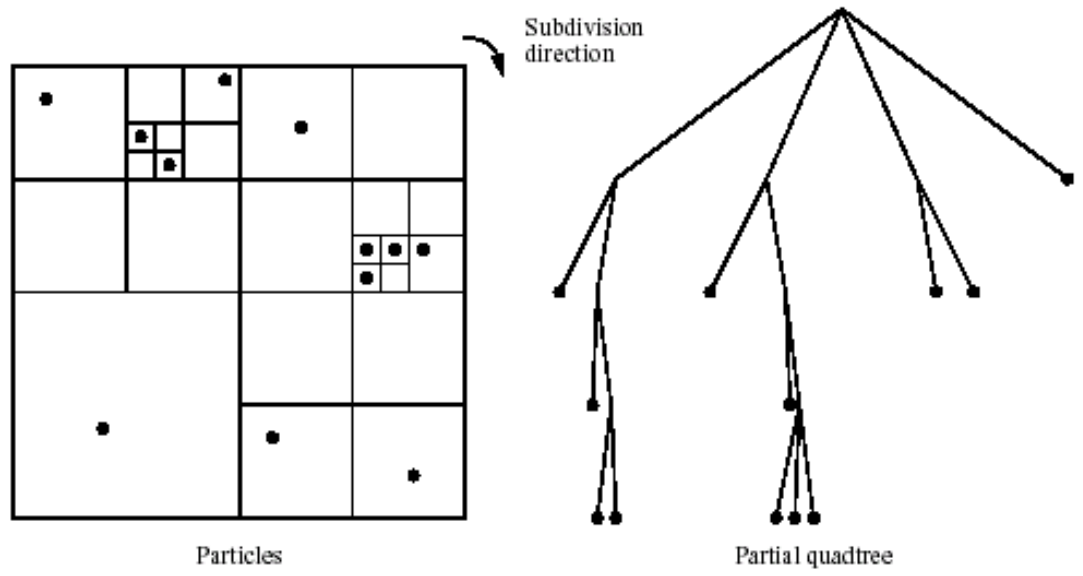


Parallel Algorithm

- The sequential algorithm is an $O(N^2)$ algorithm (for one iteration) as each of the N bodies is influenced by each of the other $N – 1$ bodies. Each of the forces on each interacting pairs are independent calculations.
- The all-pairs calculation is not feasible for most interesting N-body problems where N is very large.
- Time complexity can be reduced using observation that a cluster of bodies can be approximated as a single body of the total mass of the cluster sited at the center of mass of the cluster (coarse-graining).
- Only bodies from nearby cells need to be treated individually, and particles in distant cells can be treated as a single large particle centered at its center of mass.

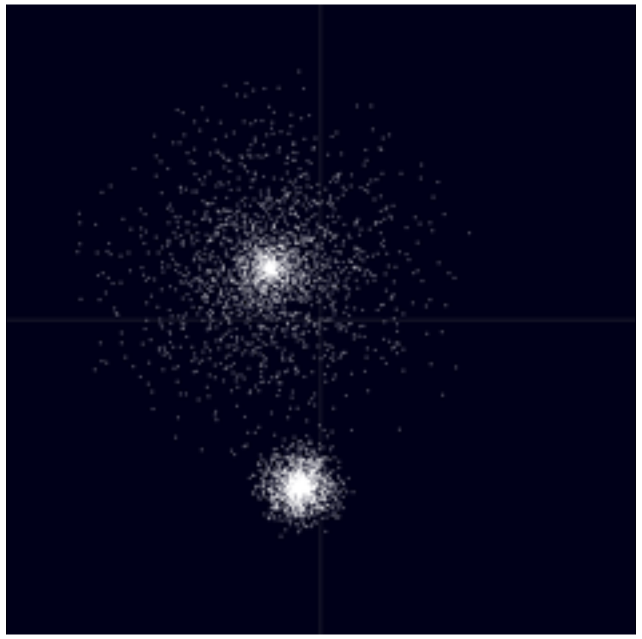
Barnes-Hut Algorithm

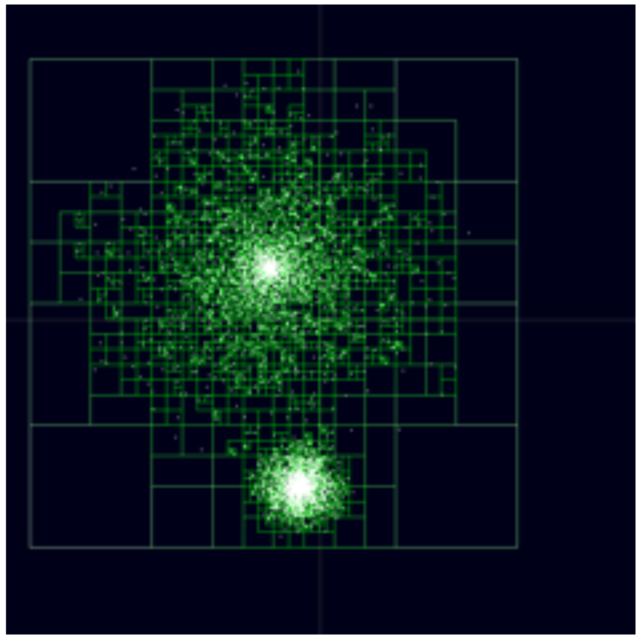
- Creates an octtree (or quadtree as shown on right) – a tree with up to eight (four) edges from each node.
  - The leaves represent cells each containing one body.
  - After the tree has been constructed, the total mass and center of mass of the subcube (subsquare) is stored at each node.
  - Force on each body obtained by traversing tree starting at root, stopping at a node when the clustering approximation can be used, e.g. when $r \leq d/\theta$ where θ is a constant typically 1.0 or less.
- Recursive division of 3D space (or 2D).
- Constructing tree requires a time of O(n log n), and so does computing all the forces, so that the overall time complexity of the method is O(n log n).

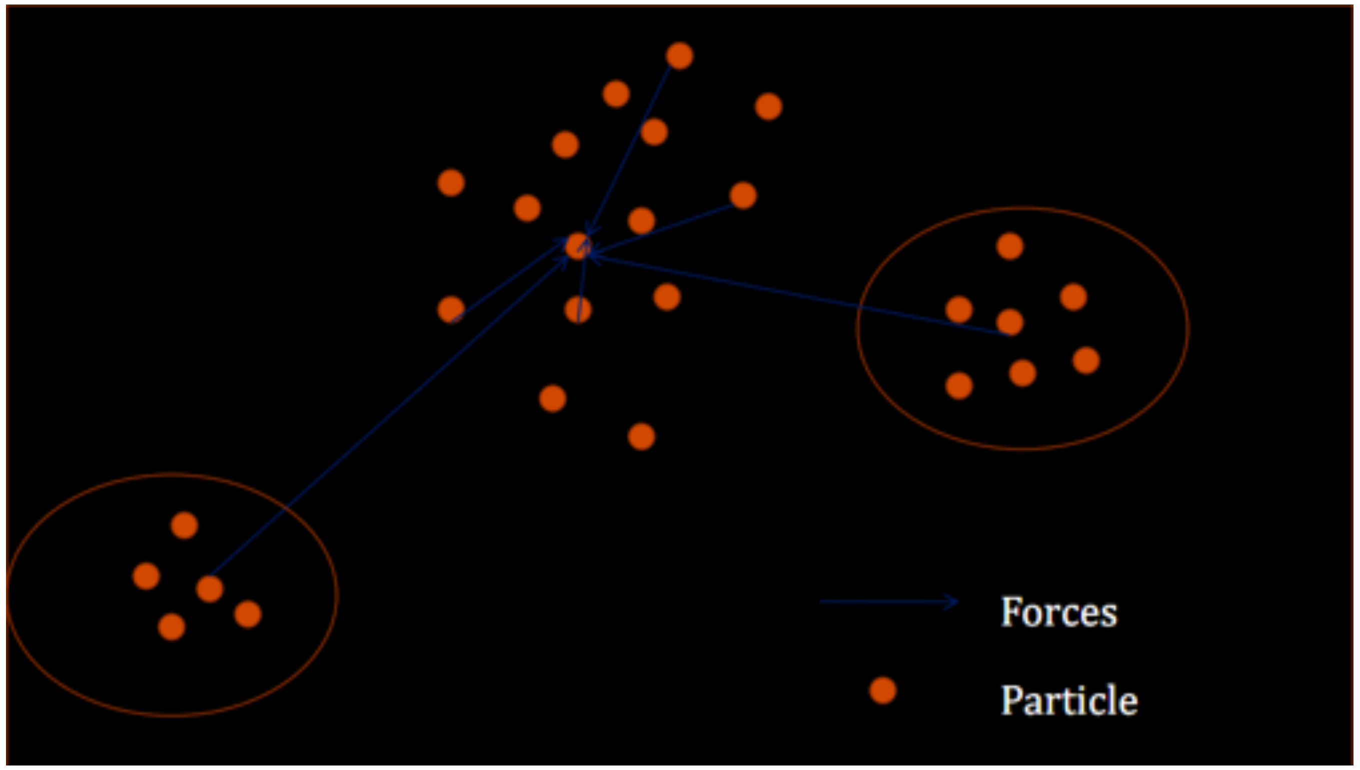


Barnes-Hut Algorithm

- Start with whole space in which one cube (square) contains the bodies.
    - First this cube (square) is divided into eight subcubes (four squares).
    - If a subcube (subsquare) contains no bodies, the subcube (subsquare) is deleted from further consideration.
    - If a subcube (subsquare) contains one body, subcube (subsquare) is retained.
    - If a subcube (subsquare) contains more than one body, it is recursively divided until every subcube (subsquare) contains one body.






Forces
Particle