UNIVERSITY OF PORTSMOUTH

N-Body Problems

HAMIDREZA KHALEGHZADEH

N-Body Problems

In this lecture we focus on the problem of *bodies moving under their mutual force of gravity* – though considerations also apply to electrostatic forces and other long-range interactions.

Important for many problems in astrophysics and cosmology - e.g. for problems of structure formation in the universe.

Isaac Newton published the solution to the 2-body problem in his *Principia* in 1687.

For three or more bodies, the only approaches available to this day are approximate or numerical solutions.

Statement of Problem

For N bodies interacting through gravitation, the acceleration due to gravity operating on particle i can be written as:

$$\boldsymbol{g}_i = \sum_{\substack{j=1 \\ j \neq i}}^{N} \frac{G m_j (\boldsymbol{x}_j - \boldsymbol{x}_i)}{\left| \boldsymbol{x}_j - \boldsymbol{x}_i \right|^3}$$

where G is Newton's constant, $\boldsymbol{x}_i$ is position of particle i, and m_i its mass.

Tow simulate the motion of particles due to gravitational interactions, we can use Euler or Runge-Kutta methods to get the positions and velocities of each particle at any given time.

General Approaches to Solution

We will discuss three general approaches

1. The “naïve” approach
2. *Particle Mesh* methods
3. *Tree algorithms* (“multipole methods”)

Naïve Approach

Force calculation takes $O(N^2)$. This typically limits use of this approach to thousands of particles

- other approaches can go to millions, or billions on a supercomputer.

Still may be preferred for its accuracy in some applications (e.g. simulating evolution of globular clusters?)

- Historically special purpose processors (GRAPE) were developed for mass computations of gravitational force law – today would presumably use GPUs?

But for simulation of more particles, need more efficient algorithms.

Particle Mesh Approaches

Rather than calculating individual force law between particles, a mathematically equivalent approach is to consider aggregate *gravitational field* produced by all particles.

The main steps in the particle mesh are:

1. Create a mesh of particles.
2. Obtain the gravitational potential field using Poisson's method,
3. Calculate the local acceleration based on the potential field,
4. Interpolate the local acceleration on the grid to find the acceleration applied on the particles.

Poisson's Equation

The gravitational potential field satisfies Poisson's equation:

$$\frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial y^2} = \rho(x, y)$$

where now ρ is the local density of gravitating matter.

If we discretize space on a grid, we can define ρ in terms of the number of source bodies near a particular point (e.g. in a cell).

We can solve this partial differential in $O(N \log N)$ time using FFTs.

Gravitational Potential

The quantity φ solved in the previous slide is the *gravitational potential field*, colloquially the spatial variation of electric "Voltage".

- Recall the discussion of the *Laplace equation* in the Parallel Programming module.

We can now define a gravitational potential as:

$$(g_x, g_y, g_z) = \left(\frac{\partial \varphi}{\partial x}, \frac{\partial \varphi}{\partial y}, \frac{\partial \varphi}{\partial z}\right)$$

Where the gravitational field, $\boldsymbol{g}$, determines the local acceleration due to gravity.

The potential and thus field can be determined by considering the distribution of all gravitating particles.

Interpolation

So far we obtain the local acceleration due to gravity in each node in the mesh

Now, we can use interpolation to calculate acceleration is applied to each particle in a node based on what cell the particle is in, and where in the cell it lies.

Particle Mesh

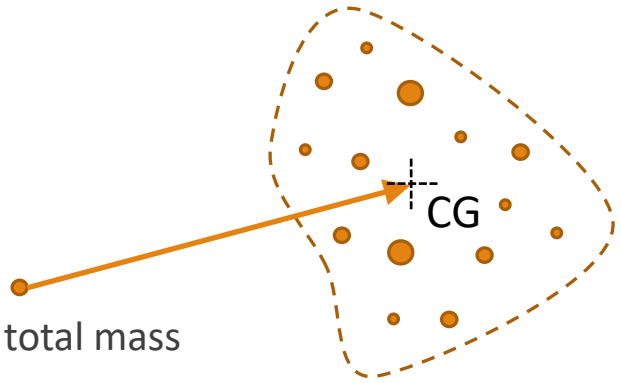
Combining these features, this means we can calculate an approximation to the gravitational acceleration on every body in $O(N \log N)$ time.

A limit here is the resolution of the mesh. This approach is not good for particles that are closer together than the mesh size.

But still this approach can be used to calculate the far-field part of gravitational force, in conjunction with other methods used for forces from the local neighbourhood.

Tree Algorithms

These all begin from the observation that often particles can be grouped together, and their gravitational effect approximated by one large mass concentrated at their *centre of gravity*.



Here M is total mass of cluster.

For spherically symmetric assemblies of matter (e.g. Earth), this approximation is exact, as originally proven by Newton, in an early application of integral calculus.

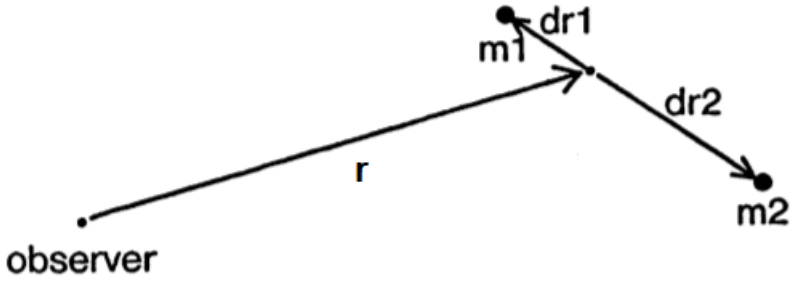
For clusters of stars (for example) that are far enough away, it can be an excellent approximation.

Monopole Formula

Consider two particles, m_1 and m_2, each no more than dr from their centre of mass. The gravitational acceleration applied upon an observer situated at a distance *r* from the centre of mass can be calculated using the monopole formula as:

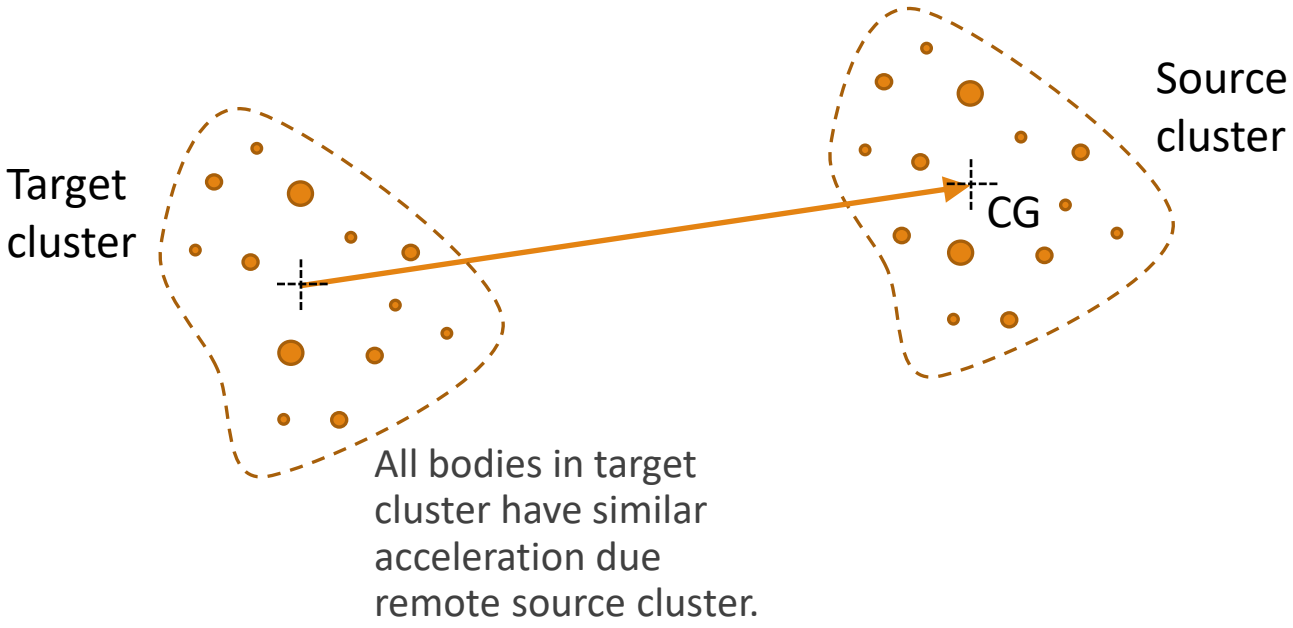
$$\boldsymbol{g} = \frac{Gm_1(r+dr_1)}{|r+dr_1|^3} + \frac{Gm_2(r+dr_2)}{|r+dr_2|^3} = \frac{G(m_1+m_2)r}{|r|^3} + O(dr^2)$$

Where dr_1 and dr_2 determines the distance between the two particles and the centre of mass respectively.



Approximating Local Field

Some of the tree algorithms go beyond clustering *sources* of gravitation, and exploit the fact that each body in a *target cluster* will have a similar acceleration due to the gravity of a far away *source cluster*:



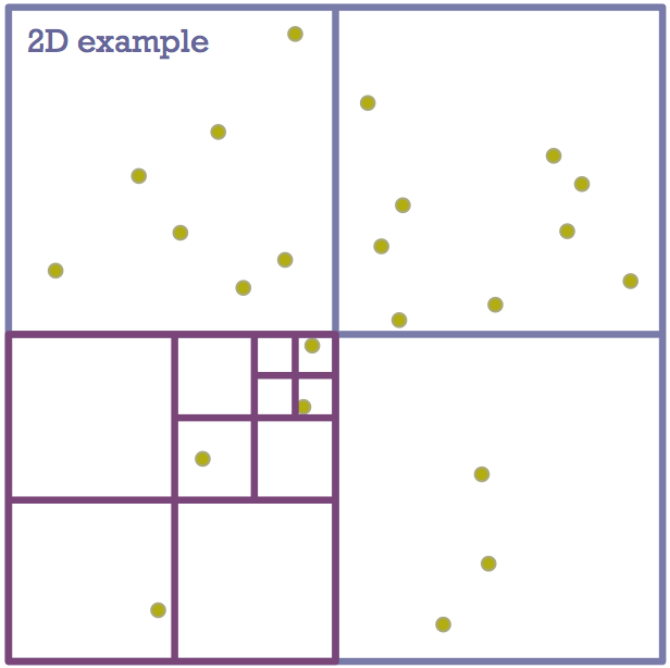
Hierarchies of Clusters

The well known algorithms that exploit these kinds of clustering are based on *hierarchies* of clusters, where at the top level we have a cluster containing *all bodies*.

This top level cluster is successively divided into smaller clusters – at each level dividing bodies of parent clusters between their daughters.

This leads to algorithms built around tree data structures, hence the common name *tree algorithms*.

Visualising Tree Algorithms



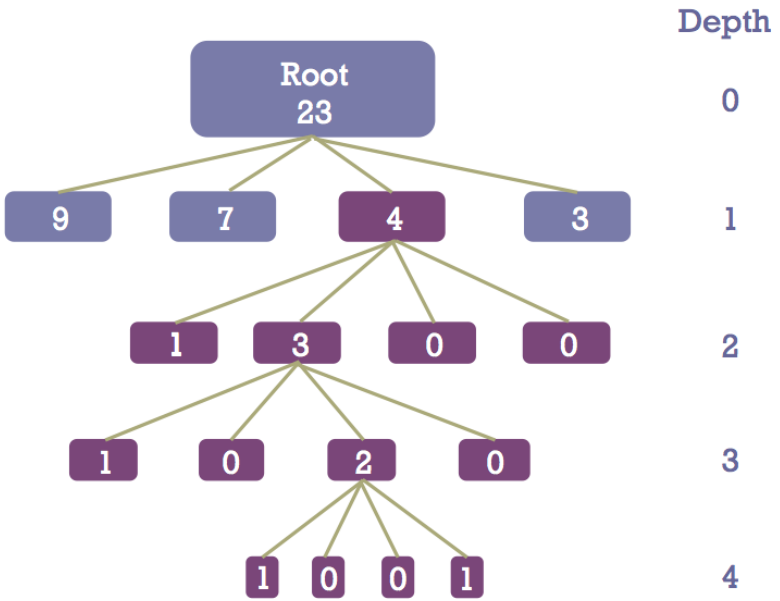


Image from http://portillo.ca/nbody/barnes-hut/

Opening of Clusters

Depending on the algorithm, *leaves* of cluster trees may either be individual bodies, or clusters containing a small constant number of bodies.

In considering forces exerted by one cluster on a body or other cluster, we can either use an averaged approximation for forces, or “open” clusters into their daughters and work recursively with the resulting smaller clusters, to get more refined evaluation of forces.

Decision whether or not to open a cluster depends on some *opening criterion*.

Well Known Tree Algorithms

All of these were developed within a few years of each other:

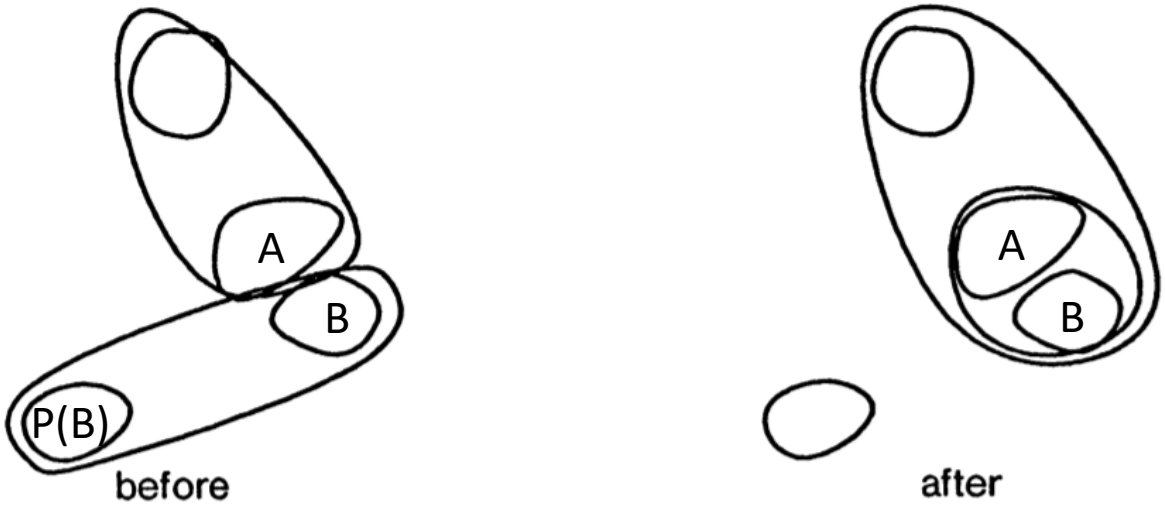
- Appel's original algorithm (1985)
- Barnes Hut algorithm (1986)

Will cover them in chronological order.

Appel's Algorithm

Based on a *binary tree* of clusters, construction based on proximities of bodies.

- Tree is updated on the fly as algorithm proceeds by a *grab algorithm*.



†Image from Appel, 1985

Opening of Appel's Algorithm

Since Appel's algorithm builds *binary* trees, It is now a simple matter to compute all of the gravitational interactions between *two clusters*.

- Uses monopole approximation for central force.

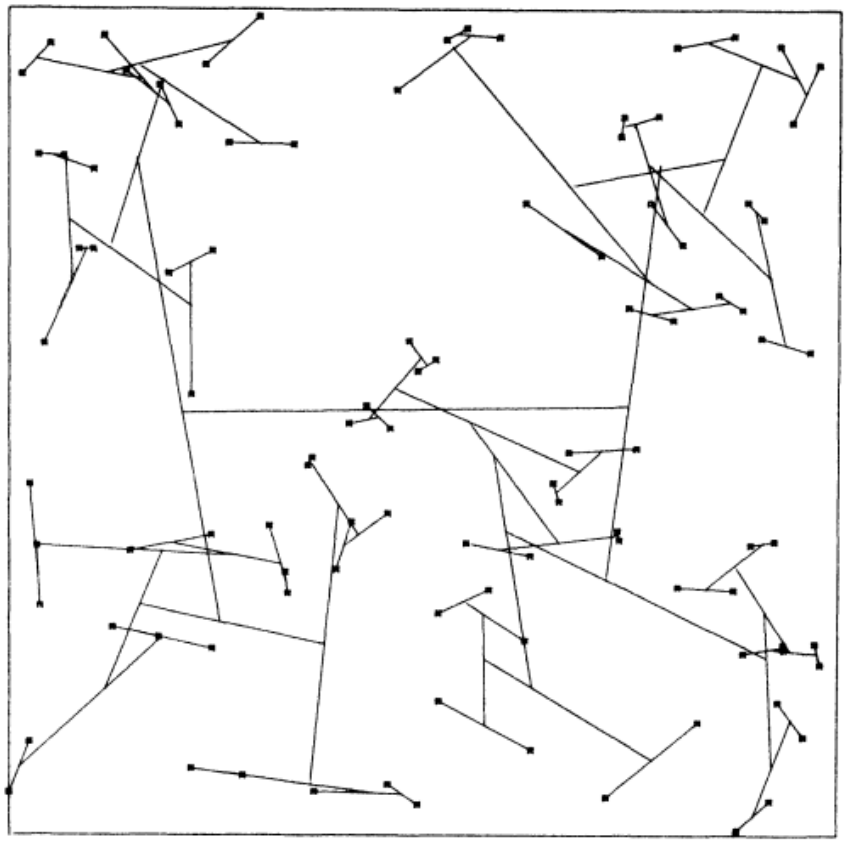
Uses "source cluster and target cluster" approach. Each cluster has an average acceleration due to clusters sufficiently far from it.

- Acceleration of daughter clusters is relative to this average acceleration of parent.

If a pair of clusters are sufficiently far apart, add central force from cluster A to average B acceleration, and vice versa

- Otherwise split the largest of the two clusters and recurse on two new pairs.

Example Appel Tree†



†Image from Appel, 1985

Barnes Hut Method

Based on clusters defined by quad-trees (2D)

- Start with a cluster containing whole of space, subdivide into 4 sub-clusters.
- Repeat as many times as needed till leaf “clusters” contain a single body.

This algorithm calculates accelerations individually for each body.

- Only particles from nearby clusters need to be treated individually.

But, particles in distant cells can be treated as a single large particle represented by a concentrated mass.

- Add central acceleration from the source cluster to the target particle

This can dramatically reduce the number of particle pair interactions that must be computed.

Two Dimensional BH Tree

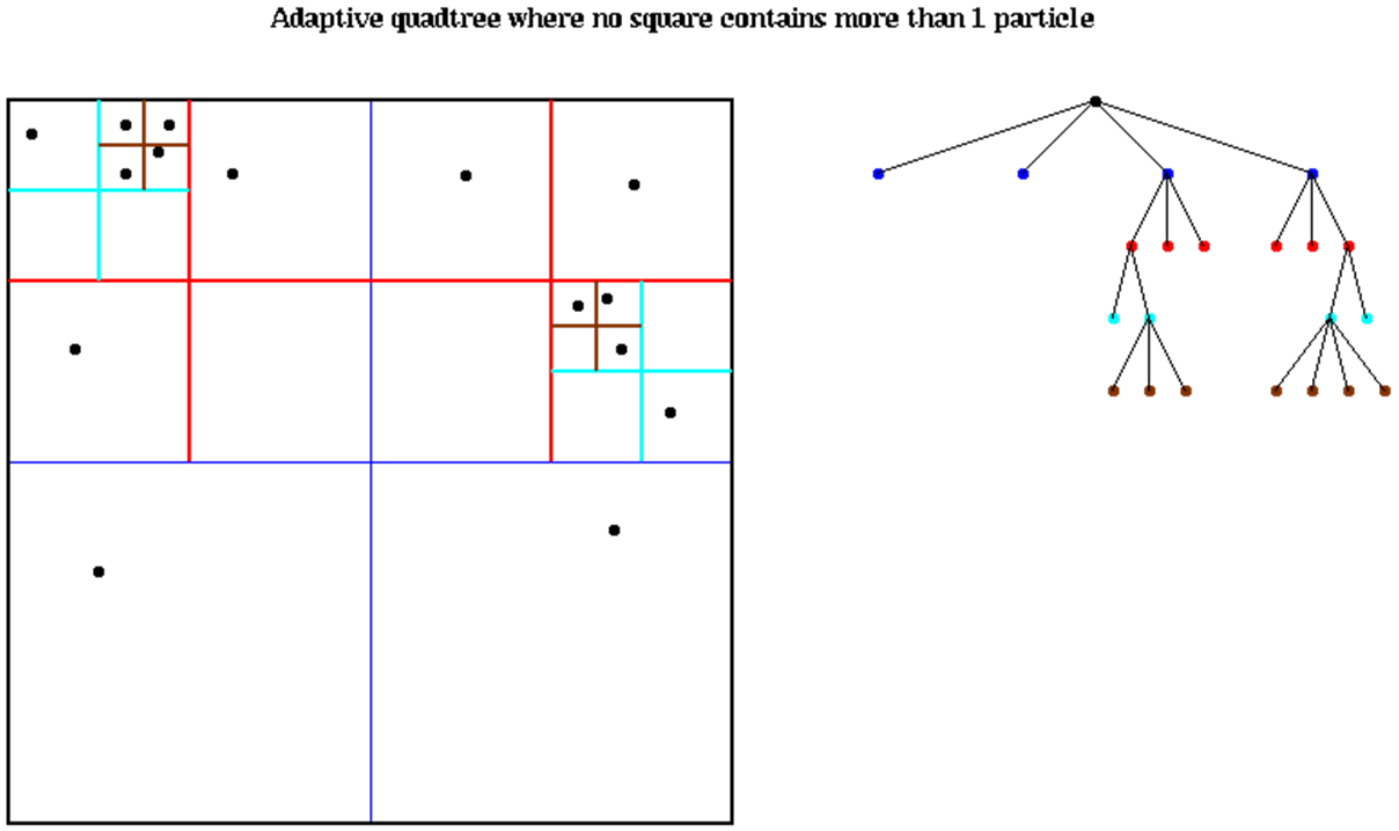
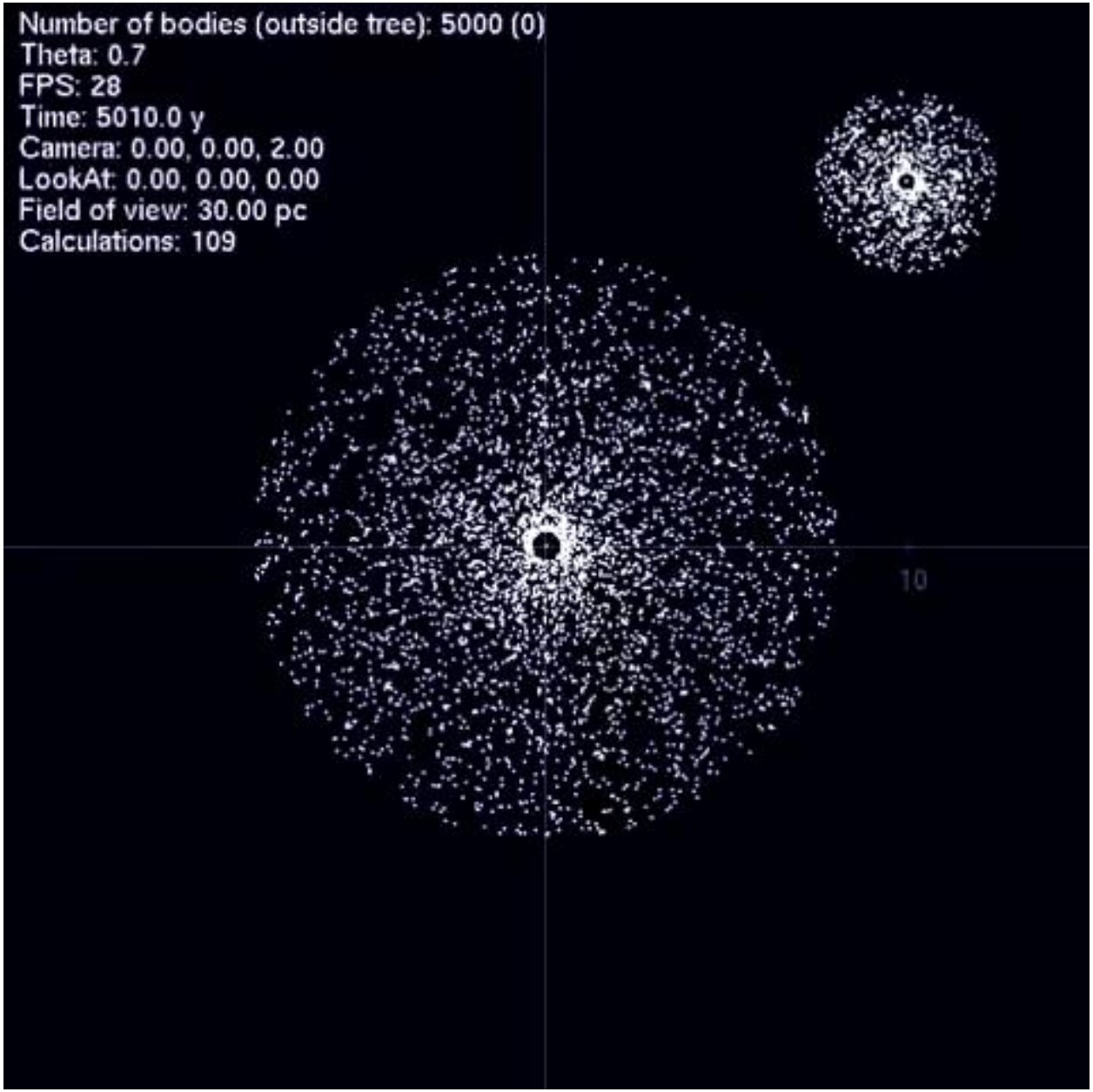


Image from www.cs.berkeley.edu/demmel/cs267/lecture26/lecture26.html



Video form Wikipedia

References

Appel, *An Efficient Program for Many-Body Simulation*, 1981.

Warren and Salmon, *A Parallel Hashed Oct-Tree N-Body Algorithm*, 1993.

Volker Springel et al, *GADGET: A code for collisionless and gas dynamical cosmological simulations*, 2001.