Programming Techniques 2024-2025

Barnes-Hut Algorithm

Hannu Parviainen

Universidad de la Laguna

October 21, 2024

Problem: Direct simulation leads to $O(N^2)$ scaling

- We need to calculate the forces between all the particles in the simulation.
- The number of force calculations depends on the number of particles, N:
  - $N = 2 \Rightarrow 1$ calculation
  - $N = 3 \Rightarrow 3$ calculations
  - $N = 4 \Rightarrow 6$ calculations
  - $N = 10 \Rightarrow 45$ calculations
  - $N = 100 \Rightarrow 4\,950$ calculations
  - $N = 1000 \Rightarrow 499\,500$ calculations
- Computational complexity: $\frac{1}{2}N(N-1) \sim N^2$
- Called $O(N^2)$ scaling.

Computational Scaling

- $T_{N=100} / T_{N=10} = 100$
- $T_{N=1000} / T_{N=100} = 100$
- $T_{N=10\,000} / T_{N=1000} = 100$
- $T_{N=10\,000} / T_{N=10} = 1\,000\,000$

Computational Scaling

- $T_{N=100} / T_{N=10} = 100$
- $T_{N=1000} / T_{N=100} = 100$
- $T_{N=10\,000} / T_{N=1000} = 100$
- $T_{N=10\,000} / T_{N=10} = 1\,000\,000$
- **Not good!**

Computational Scaling

- $T_{N=100}/T_{N=10} = 100$
- $T_{N=1000}/T_{N=100} = 100$
- $T_{N=10\,000}/T_{N=1000} = 100$
- $T_{N=10\,000}/T_{N=10} = 1\,000\,000$

- **Not good at all!**

What can we do?

- Find an algorithm that scales better.
- **Barnes-Hut Algorithm (Nature, V. 324 1986)**
- Scales as $O(N \log N)$

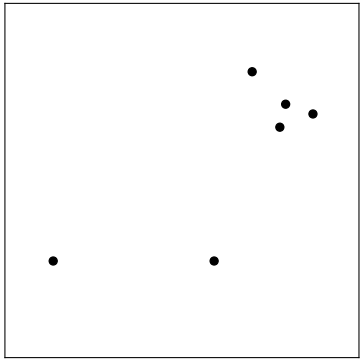
- $T_{N=100}/T_{N=10} = 20$
- $T_{N=1000}/T_{N=100} = 15$
- $T_{N=10\,000}/T_{N=1000} = 13$
- $T_{N=10\,000}/T_{N=10} = 4000$

How?

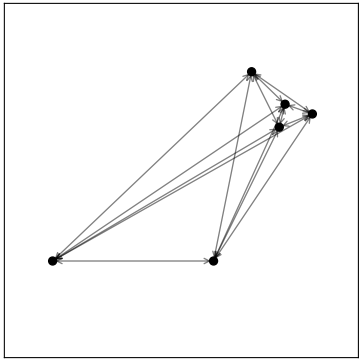
How?

Approximate far-away clumps of particles as point masses.

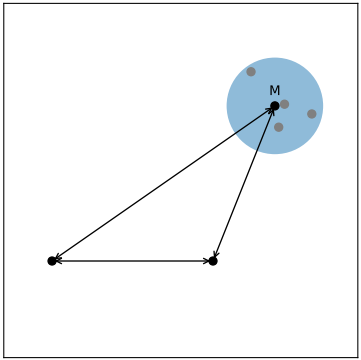
Barnes-Hut



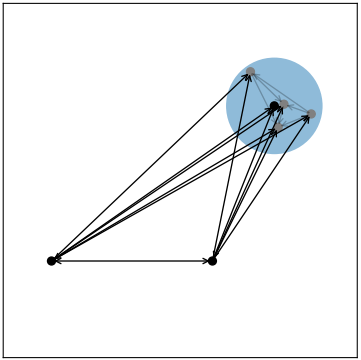
Barnes-Hut



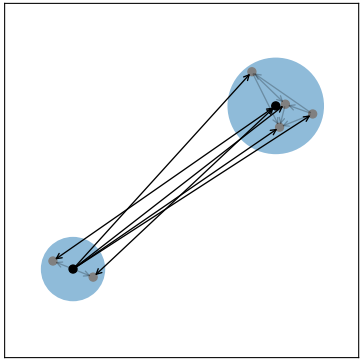
Barnes-Hut



Barnes-Hut



Barnes-Hut



How?

Use a spatial data structure to organise the particles.

- **Octree:**
  - A tree data structure that divides 3D space into hierarchical cubes.
  - A 3D version of a quadtree.
- **Octree node:**
  - An octree node represents a cube in 3D space.
  - Each node can be subdivided into eight smaller cubes (children nodes).
  - A node has a link to its parent node.
  - In our case, a node can contain:
    - A single particle (either a pointer or an array index).
    - Total mass of all the particles in the descendant nodes.
    - 3D center of mass.

Octree Visualization

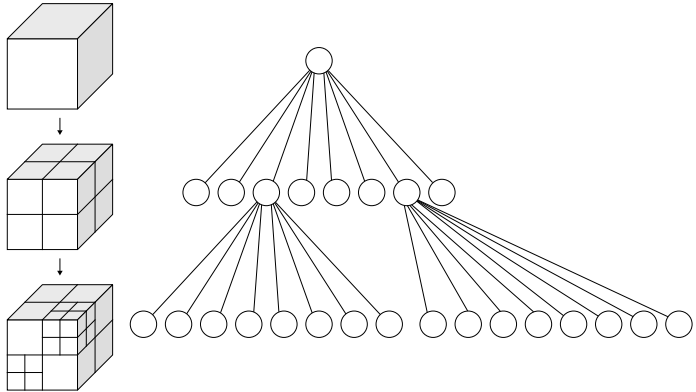
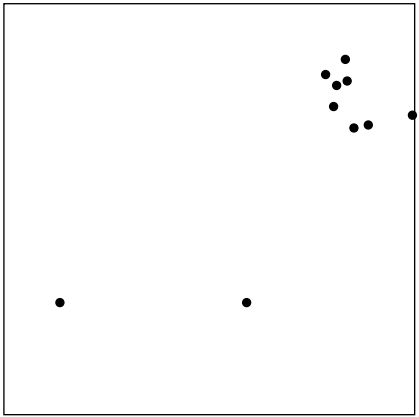
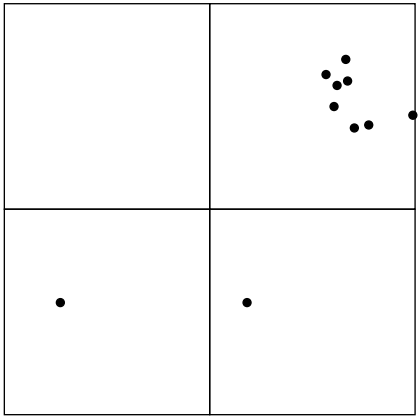


Image by: WhiteTimberwolf: CC BY-SA 3.0

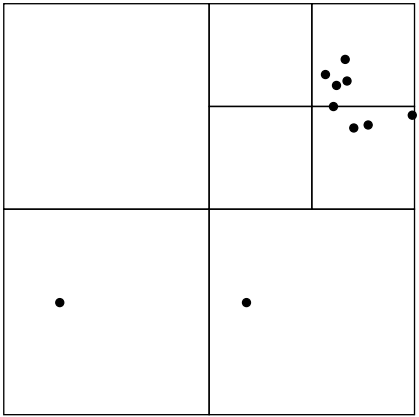
Octree Visualization



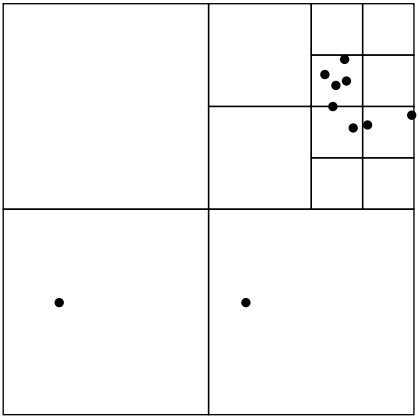
Octree Visualization



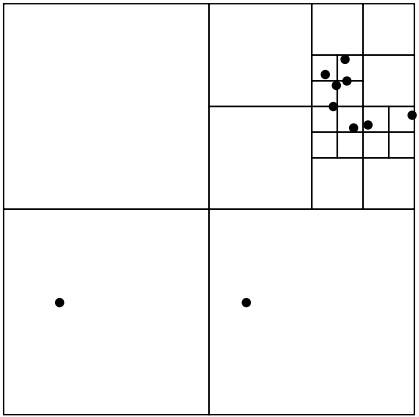
Octree Visualization



Octree Visualization

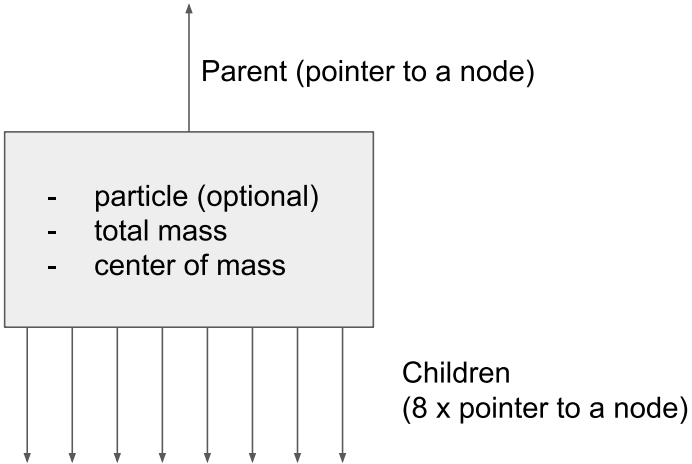


Octree Visualization



Octree Node Structure

Node:



Building an Octree

- Find the node the particle belongs to.
- If the node is empty, assign the particle to the node.
- If the node already contains a particle:
  - Subdivide the node.
  - Move the existing particle to the correct child node.
  - Try to assign the new particle to the correct child node.
  - If the child node already has a particle, subdivide again.
- Recursive subdivision makes this process clean and efficient.