Many, many particles

Ferienakademie 2015:
Let's play! Simulated Physics for Interactive Games

Erik Wannerberg
Fakulät für Informatik
Technische Universität München
Email: e.wannerberg@tum.de

November 30, 2015

1 Introduction and motivation

Systems with many particles are of high importance in many computational areas, and are not only of academical interest, but also play a big role in many topics in industry. These range all the way from simulation of the stars in the universe – which can be modelled as very, very many particles – where long-range forces have to be added up for each of them, to fluid simulations using particle-based methods, such as Smoothed Particle Hydrodynamics (SPH) [1] or Molecular Dynamics (MD) [2], where the forces are short-ranged, but the particles are as many.

However, they are also used for visual effects in animations in both movies and games [3, 4], where a cloud of smoke or an explosion can be visualized by the blurring of a large group of point-like particles. These can then easily be made to move, simulating a wind, and can for example be born from a fire with a limited lifetime, to simulate dissipating smoke, creating very realistic effect by making the distribution and recreation over time a pseudorandom function. Drawing the whole trajectory of such a particle can also create a realistic model of hair or grass (see Figure 1 for an illustration).

As already mentioned, these simulations also typically include *interactions* between these large numbers of particles – in the physics simulations, these are the interaction forces. Also, in movies and games, although the visualisation and simulation might not require the physical forces per se in order to create a realistic effect, there is typically a need to check that the particles do not pass through static parts of the scene, such as walls, or other dynamic objects, such as people, cars or aliens.

In both of these cases, if the forces or possible collisions are calculated once for each object with every other object, as the simplest algorithm would suggest, we would end up with a number of individual calculations of order $\mathcal{O}\left(N^2\right)$ for N objects. For a reasonably

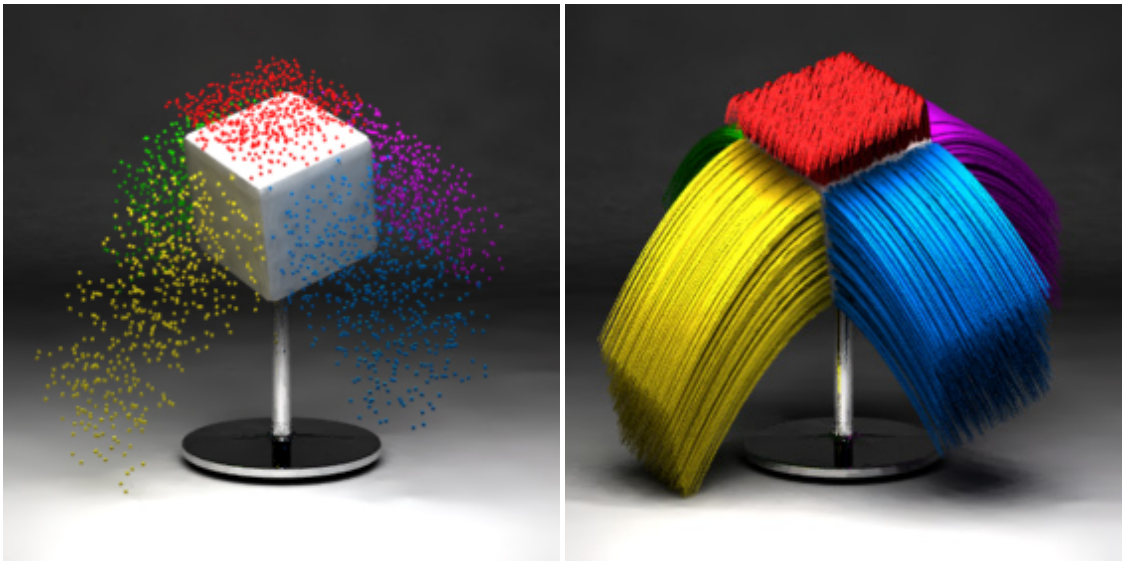

Figure 1: An example of the illustrative effects of particles, using the 3D rendering software *Blender* [5]. Particles are emitted with a certain speed from the surfaces of a cube with a pseudorandom ditribution in space and distribution time, and are pulled downwards as by gravity. To the left, the particles are visualised as points, not unlike coloured snow. To the right, the whole particle paths are drawn at once, creating a hair-like impression. The simulation backend of Blender is the *Bullet* library [6], which implements several different broad-phase collision culling strategies, among them Bounding Volume Hierarchies. Picture from [4]

low computational time, this which is very much too high when the number of particles grows large.

Beyond this introduction to uses of many-particle systems, the purpose of this report is therefore to introduce some algorithms that help reduce the computational time. First, in section 2, solutions for long-distance force calculations are presented, and in section 3, some solutions for more local interactions are presented, followed by a short summary in section 4.

2 Solutions for long distance interactions

In physics, there are forces that diminish so slowly with increasing distance, that even across distances as large as that between clusters of galaxies, they still remain non-negligible. Therefore, when doing simulations on such things as the evolution of the universe from Big Bang until now, one need to include force calculations between practically *all* stars. However, by grouping together force contributions of distant objects, one can do this without needing to calculate interactions for each pair of objects separately. The *Barnes-Hut* algorithm and the *Fast Multipole* both implement this strategy to varying degrees, and are described below.

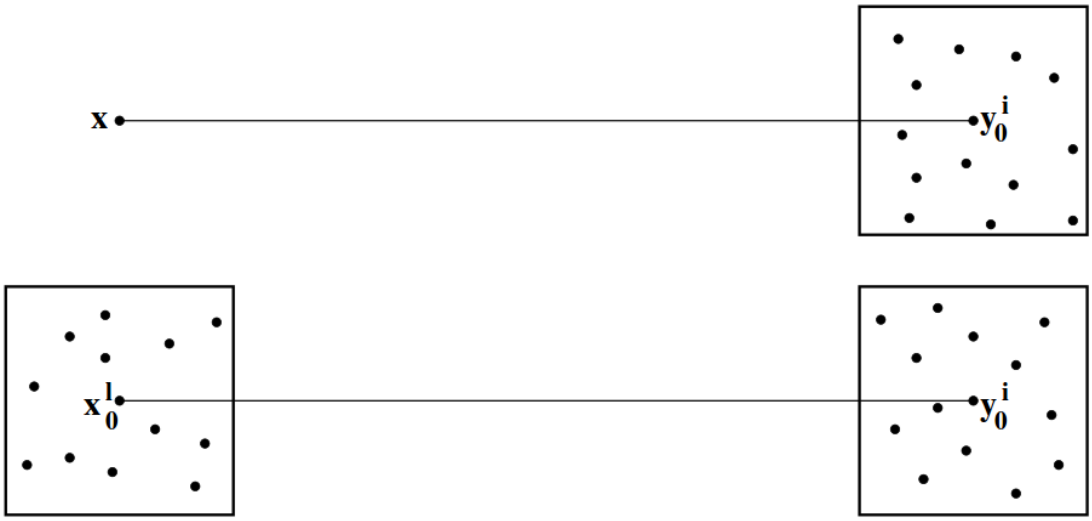

Figure 2: An illustrative comparison between the Barnes-Hut method (above) and Fast Multipole methods (below). In the Barnes-Hut method, forces are calculated on each particle (left) by combining the contributions from a cluster of particles (right), that are sufficiently far away. In Fast Multipole methods, forces on groups of particles (left) are additionally only calculated once. Figure from [7].

2.1 Barnes-Hut

The Barnes-Hut algorithm combines the interaction force from a group of particles *sufficiently far away* in comparison to the group's extent into the force from a *pseudoparticle* at the position of their center of mass. The relationship between distance and size determines both the accuracy and computational cost of the method. and is controlled by varying a parameter θ, such that one only uses the pseudoparticle at a distance r for the force calculation if $r > l/\theta$, where l is a length related to the size of the area, from which particles have been included in the pseudoparticle (for example the diameter of that area). By building up a hierarchical system of pseudoparticles once, the cost of one force calculation on one particle (in a system of N particles in D dimensions) drops from $\mathcal{O}(N)$ to $\mathcal{O}\left(\theta^{-D} N \log N\right)$. This gives the Barnes-Hut algorithm a computational complexity $\mathcal{O}\left(\theta^{-D} N \log N\right)$ [7, 8].

2.2 Fast Multipole methods

Fast Multipole methods extend the Barnes-Hut idea in that they also summarize the interaction *received* by a group of particles, using more advanced mathematical methods to then distribute this information to members of the receiving group. This reduces the complexity in particle number N to $\mathcal{O}(N)$ [7], although with a rather large constant base runtime, since the hierarchical systems necessary to distribute the force to the group members are quite expensive to set up. An illustrative comparison between the two methods can be found in Figure 2.

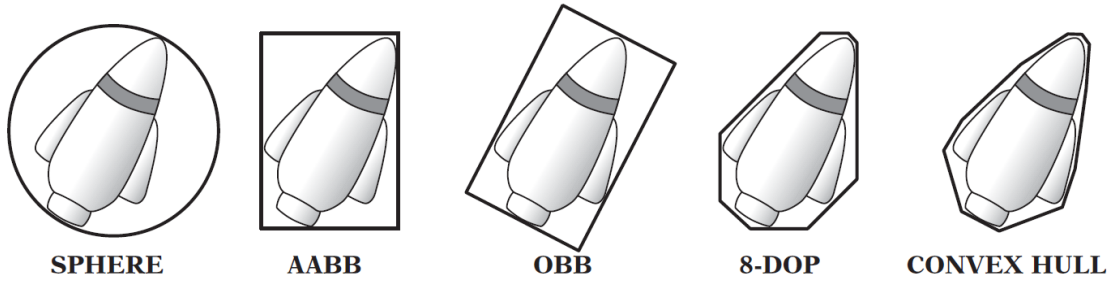

Figure 3: Different types of bounding volumes in 2D. From left to right, the cost of computing both the bounding box and the intersection test increases, but the fit to the figure also gets better, meaning more false intersections are excluded. The abbreviations AABB for *Axis-Aligned Bounding Box* (second to left), OBB for *Oriented Bounding Box* (third to left) and DOP for *Discrete Oriented Polytope* (fourth to the left) are standard in literature. Figure from [9].

3 Solutions for short distance-only interactions

With shorter range interactions, such as short-range forces (in MD and SPH) or collisions (in any simulation), it is often enough to only calculate the interactions between particles sufficiently close to each other, something often referred to as *culling*. Since this removes many calculations, it can reduce up the total computational time significantly. Here there are also many different methods, which typically either divide the *space* in which objects are considered for interaction, called *spatial partitioning*, or by dividing the group of *objects* into separate subgroups that are compared against each other for interaction, as is done in *Bounding Volume* methods. Briefly described below is the *Linked cell-list* algorithm, a uniform spatial partitioning scheme, and the ideas behind using *Bounding Volume* hierarchies.

The topics of spatial partitioning (including hierarchical methods) and bounding volumes are explored considerably deeper in [9]. Here, one finds a thorough introduction to the theory and practice, discussions of optimisations, and other uses of the schemes besides collision tests, among much else.

3.1 Spatial partitioning: Linked cell-list

The *linked cell-list* algorithm divides the region into cells of uniform size. For force calculations, this size is related to the cut-off radius, that tells after which distance the force contribution can be neglected, and for collisions, it is often related to object size. In the algorithm, the interactions are then only calculated between objects or particles in cells within the cut-off radius of each other (usually the neighbouring cells) or within each cell, in the case of collisions. With constant density of uniform-sized objects, the complexity is of order $\mathcal{O}(N)$ [7]. However, there are problems with this method when the objects are of very different sizes, as either one object can be in very many cells, if it is much bigger than the cells, or there can be very many objects for each cell, if they are much smaller than the cell, leading to computational overhead [9].

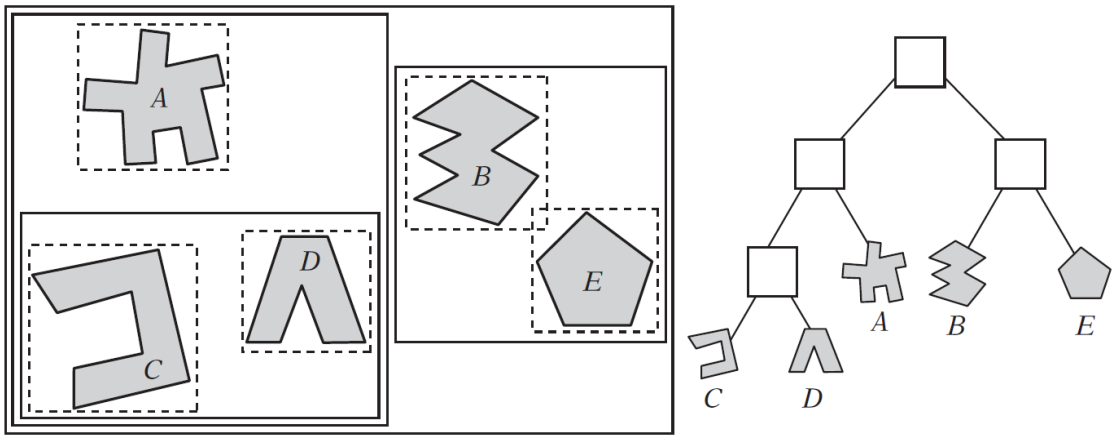

Figure 4: An example bounding volume hierarchy for collision testing. Left: The spatial layout of the five objects (gray) as well as levels of bounding boxes (dashed and whole-drawn rectangles). Right: The hierarchy as a tree diagram. Here, for example, the intersections between groups A-C-D and B-E only needs to be considered through their bounding boxes once, as they do not overlap, instead of six individual tests. Figure from [9].

3.2 Bounding Volumes

Computing whether two objects collide or not is typically done by checking whether the two objects intersect anywhere. This can be a computationally quite expensive operation, especially for curved objects and other complex geometries. By wrapping a complex object in one of simpler geometry, a bounding volume, one exclude some test by first excluding the cases in which the wrapping geometries do not collide, meaning objects are definitely not colliding, called *narrow-phase culling*. The wrapping geometries can be for example rectangular boxes or spheres (circles in 2D), for which intersection tests are very cheap (see Figure 3 for more examples).

3.3 Bounding Volume hierarchies

By making a tree-like structure of bounding volume *hierarchies* enclosed within each other, one can not only make the comparisons between objects simpler, but also use them to exclude interactions between whole groups of separated objects. This process, illustrated for clarity in Figure 4, is termed *broad-phase culling*, and is similar to the hierarchical methods of Barnes-Hut and Fast Multipole in section 2. With these approaches, the number of interactions between N objects of arbitrary sizes and positions can be reduced to $\mathcal{O}(N \log N)$, and typically even less. This however requires careful consideration of the tree structure, build-up and traversal.

4 Summary

There are many applications in which systems with very many particles are used with great success, in both scientific simulations using the methods of Molecular Dynamics or Smoothed Particle Hydrodynamics, or in particle effects in movies and games. The usage of these large systems of particles requires usage of specialized algorithms to be computationally viable however, where one reduces the computational effort of interactions between the particles. For long-range force calculations, this is done by summarizing forces once for larger groups of particles, as in the Barnes-Hut and Fast Multipole methods, to not have to compute each pair force. For shorter-range forces or collision tests, the computational effort is reduced by excluding negligible interactions and collisions between very distant objects, either by dividing the space in spatial partitioning methods such as the Linked cell-list algorithm, or by dividing the set of objects into sets that are more or less likely to collide, by wrapping them in hierarchies of Bounding Volumes which tests are done on.

References

[1] Joe J Monaghan. Smoothed particle hydrodynamics. *Annual review of astronomy and astrophysics*, 30:543–574, 1992.

[2] Wikipedia. Molecular dynamics — wikipedia, the free encyclopedia, 2015. [Online; accessed 26-November-2015].

[3] W. T. Reeves. Particle systems—a technique for modeling a class of fuzzy objects. *ACM Trans. Graph.*, 2(2):91–108, April 1983.

[4] Wikipedia. Particle system — wikipedia, the free encyclopedia, 2015. [Online; accessed 18-September-2015].

[5] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Blender Institute, Amsterdam, 2015.

[6] Erwin Coumans and et al. Bullet physics library. `http://bulletphysics.org/`, 2015. [Online; accessed 17-November-2015].

[7] Michael Bader. Lecture notes: Scientific computing ii; molecular dynamics (force computation: Linked cell, barnes-hut, fast multipole). `http://www5.in.tum.de/lehre/vorlesungen/sci_compII/ss15/moldyn_03.pdf`, 2015. [Online; accessed 20-July-2015].

[8] Josh Barnes and Piet Hut. A hierarchical $\mathcal{O}(N \log N)$ force-calculation algorithm. 1986.

[9] Christer Ericson. *Real-time collision detection*. CRC Press, 2004.