# Simulating Particle Interactions

Darian Hall
University of Colorado Boulder

April 22, 2019

In the modeling of particles at a subatomic particle, it is necessary to consider countless interactions with surrounding particles. The core of this process invokes a need for algorithms that are both time efficient as well as spatially efficient for large quantities of particles to be simulated at the same time.

## 1 Data Structures

The need for a data structure that can efficiently search through n particles and handle collisions with all other particles inspired the use of an octree (figure 1). Ordinarily, to naively search for collisions with all other particles being actively updated is an $O(n^2)$ process. Naturally such a costly process is of little value and serves as a massive bottleneck for the performance of any program simulating n-bodies. By introducing an octree, the algorithm complex is reduced to a much more desirable $O(n \log n)$ run time. This is made possible by utilizing a variant of the octree structure, commonly referred to as the Barnes-Hut tree. In this structure, each node retains a pointer to some type of data to be stored in the tree and 8 octree children pointers. The structure subdivides recursively, only keeping data in the leaf nodes.

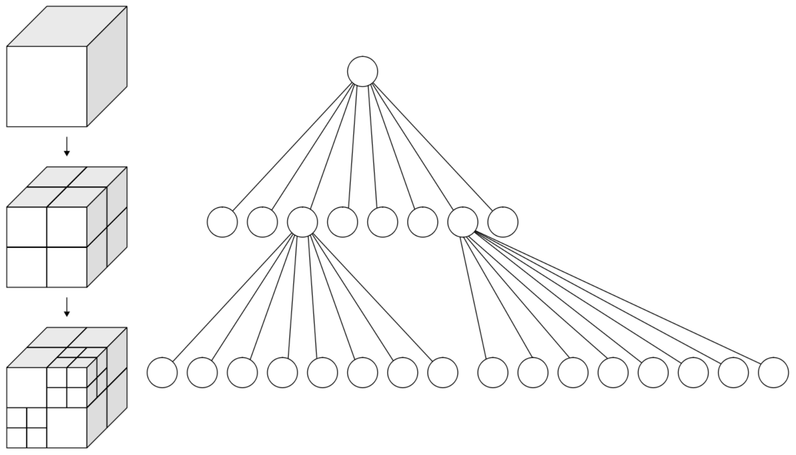

Figure 1. Octree subdividing each time a point is inserted into the tree.

With the octree, it is possibly to quickly query any volume surrounding a point. Such an ability is exactly as desired, providing the ability to search a small area around each particle for collisions, rather than searching every single particle for all n particles.

While the octree is the driving force behind the particle simulation, it is not the only useful structure. Another useful structure that was created is an event queue (min heap) that prioritizes the particles that were stored to the queue first. To keep particles that interact together, the queue creates a tuple of two particles with a collision time found when updating the octree. Particles are stored to the event queue until the tree is done checking for interactions, whereupon they are popped from queue and their decay products are stored for later purposes.

The last structure is somewhat less important to the simulation, but allows for the creation of n particles of some type with a normal distribution for both the velocities and positions. The structure generates an array of particles with given specifications to be stored and updated in the octree.

## 2 Methodology

The basic pipeline to the particle simulation is as follows. The user specifies how many particles of a certain type they would like to make and those are stored in an array that is passed to a vector pointing to the array. This allows for the creation of multiple different types of particles. The vector of particle objects is then passed to the octree where the octree keeps a pointer to the particle objects but does not instantiate any new objects. Once all the particles are stored in the tree, the particles are updated and any possible collisions are checked. Particles that collide are removed from the tree and pushed into the event queue where decay products are then processed and returned to the main for the user to print or write the decay products to a file of their choice. This process repeats until the user decides to stop the simulation.

## 3 Results

Putting the code together, the program is able to calculate all collisions that occur between approximately 20,000 particles updating the tree 500 times in the span of 5 or so minutes. This is a rather impressive feat since the traditional method of checking for collisions has a complexity of $O(20000^2) = O(400000000)$ whereas the Barnes-Hut method has a complexity of $O(20000 * \log(20000)) \approx 86000$. Such is a reduction in complexity is greatly appreciated and shows the importance of choosing data structures in the realm of n-body simulations.

## References

Barnes, Josh, and Piet Hut. “A Hierarchical O(N Log N) Force-Calculation Algorithm.” Nature, 4 Dec. 1986, www.nature.com/articles/324446a0.