N-Body Problem and Barnes-Hut Algorithm

by Martino Ferrari

1 N-Body

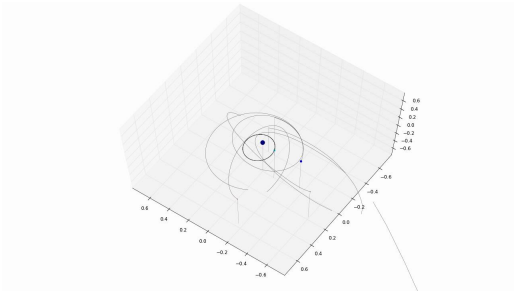

Figure 1. Classical *N-Body* problem

The *N-Body* problem is a classical astronomical problem to predict the motion of N celestial objects. With only 2 bodies the problem is completely solved, however with 3 or more bodies there is no exact solution and only numerical simulation can give us a prediction of the motion.

The motion of the bodies is driven by the gravitational interactions between them, the gravitational force is given by the Newton's law of universal gravitation:

$$F = G\frac{m_1 m_2}{r^2}$$

As from the second Newton's motion low we know that:

$$F = m \cdot a$$

We can derive the acceleration of each gravitational interaction, than is possible to integrate the acceleration twice to get the position of body at a certain time instant.

However to be able to compute the interactions of the N bodies we need to compute the acceleration N^2 times, or with simple optimizations $\frac{N(N-1)}{2}$ times (still of order $\mathcal{O}(n^2)$).

For this reason for big number of bodies there is the need of using more sophisticated algorithms.

2 Barnes-Hut

The *Barnes-Hut* simulation is an approximation of the *N-Body* simulation that permits to perform simulation with very similar accuracy with a an order of $\mathcal{O}(n \log(n))$ instead of $\mathcal{O}(n^2)$. This improvement is reached by the subdivision of the space in sub-quadrant organized as a tree.

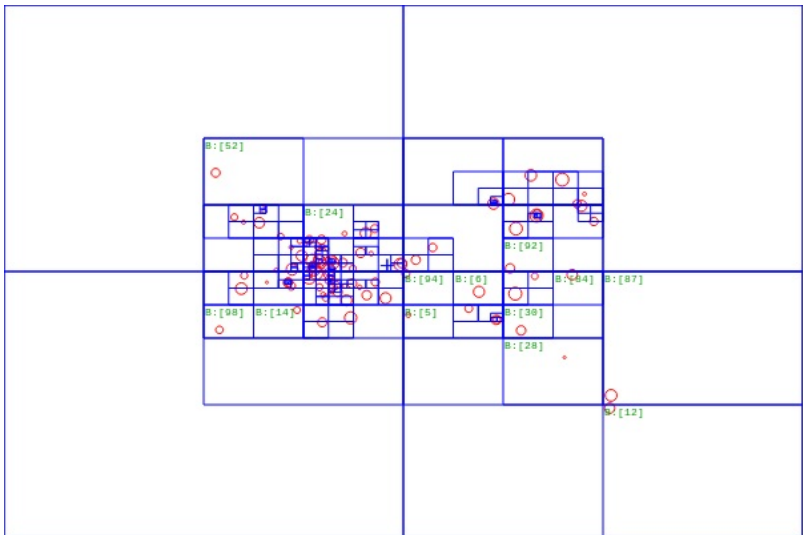

Figure 2. 2D *Barnes-Hut* space division

As the gravitational force has a quadratic dependence to the distance of the bodies, far bodies have weak interaction and so it's possible to approximate the interaction of far away areas by just taking in account their average mass and gravity center.

Using a tree like structure is possible as well to have different granularity in different areas of the simulation as well as choosing the right level of granularity depending from the distance.

3 Implementation

The implementation of the 3D version of the code was straight forward, first adding the z coordinate to the node position and momentum, then dividing the space in 8 instead of 4, finally initializing the position of the bodies taking in account the new coordinate.

The implementation of the brute force algorithm is also very simple as it's just a double for-loop.

I choose to implement the simulation also in *C++* (only in 2D) using *SDL2* and a custom 2D vector implementation, the result in performance is incredibly better then the python one (around 100 *fps* real-time rendering even with the brute force algorithm).

4 Results

The first test was to check if the implementation was valid, after 500 iteration the first body z coordinate was actually at 0.557 as expected.

The simulation of the galaxy was very slow and I interrupted after 5060 iterations, in the following figure it's possible to see the evolution of the galaxy in 5000 steps (a *gif* file animation available).

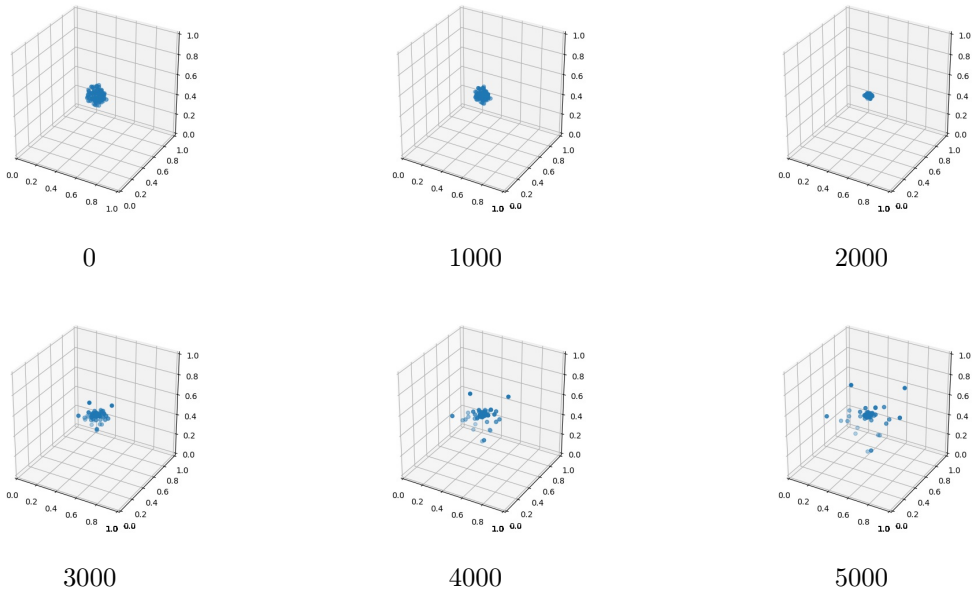

Figure 3. First 5000 iterations for the 3D *N-Body* simulation

Then we were asked to analyze the difference in performance between the Barnes-Hut method and the brute force one. To do so I choose to variate the number of bodies from 10 to 1000 (with varying interval) and simulate the first 100 iteration and plotting the average iteration time (numerical results are available in the *csv* files attached).

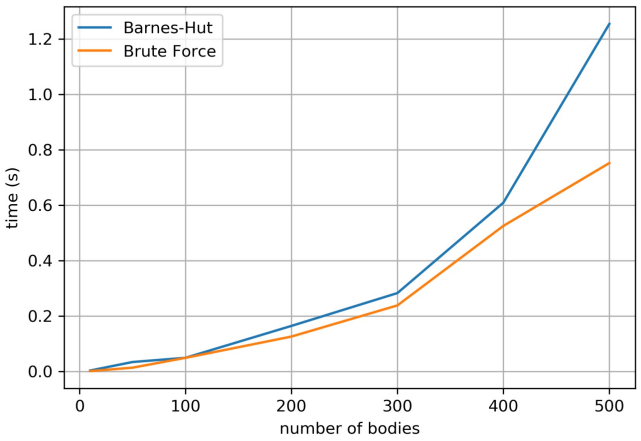

Figure 4. Difference in performance of the two algorithms

In the plot in figure 4 is possible to see how for small number er of bodies the *brute force* algorithm perform better of the *Barnes-Hut* one as the overhead of the tree creation is bigger of the advantages that it gives. However the tendency change and for 1000 bodies the *Barnes-Hut* algorithm perform better, as shown in figure 5.

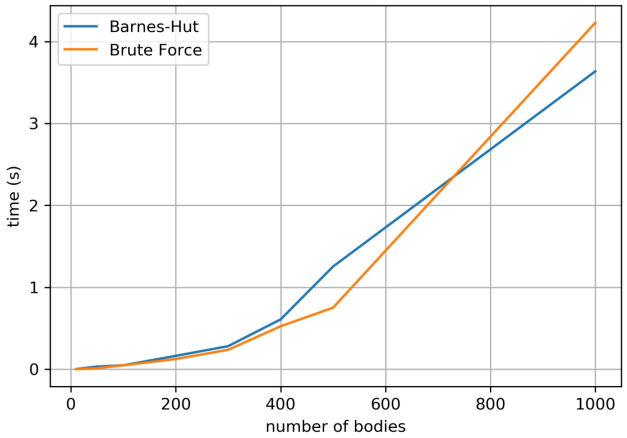

Figure 5. Difference in performance of the two algorithms

The performance of the *Barnes-Hut* algorithm depends also from the distribution of the bodies in the space, as if the bodies are very close it will not be possible to take advantage of the spatial tree approximation.

Changing the initial condition of the simulation to a bigger radius (1.0) and faster initial speed (1.0) the results for the *Barnes-Hut* are much more promising (as expected), while of course no changes are detected for the *brute force* one:

Number of bodies	Method	Initial radius	Initial speed	Average time
100	BH	0.1	0.1	0.049 s
100	BF	0.1	0.1	0.048 s
100	BH	1.0	1.0	0.017 s
200	BH	0.1	0.1	0.164 s
200	BF	0.1	0.1	0.125 s
200	BH	1.0	1.0	0.056 s

Table 1. Difference in performance depending from initial condition

To improve the performance it would probably possible to implement a mixed simulator that choose the method to simulate depending on few condition (like sparsity of the bodies, number of bodies etc).