VISUALIZATION SYSTEM FOR SIMULATION OF DYNAMIC MORPHOLOGY OF HONEYBEES

Ashwin Sankaralingam Professor Orit Peleg
Department of Computer Science, University of Colorado Boulder

1 INTRODUCTION

Honeybee forms a swarm of itself, with various honey bees attached to one another. The structure that is formed is stable to external physical environment (wind, displacement and temperature). A small change in position of the honeybee affects the swarm completely, therefore suggesting it to be a molecular dynamic system. We use particle-based simulations of a passive assemblage to suggest a behavioral hypothesis that individual bee respond to local variations in strain and spring forces.

The colonies shows an inherent ability to cope with ever changing environment and adapt its position for survival. The clusters tune their surface area, to prevent the colonies from getting damaged.

During the process of creating the simulation, various optimization techniques were tried in order to reduce the time of rendering. Molecular dynamics (MD) is a computer simulation method for studying the physical movements of atoms and molecules. The atoms and molecules are allowed to interact for a fixed period of time, giving a view of the dynamic evolution of the system. In the most common version, the trajectories of atoms and molecules are determined by numerically solving Newton's equations of motion for a system of interacting particles, where forces between the particles and their potential energies are often calculated using interatomic potentials or molecular mechanic force fields.

For the scope of project, we chose to work on Microcanonical ensemble, the method where system operates independent of temperature. Also, The potential energy function U(X) of the system is a function of the particle coordinates X. For every time step, each particle's position X and velocity V may be integrated with a symplectic integrator method such as Verlet integration.

The major goal of the study is to create a visualization system for the simulation of how honeybees would move given a physical force along a particular axis. The system was to have protocols to change the threshold for various interacting forces. Apart from this, the study also focused on understanding the collective dynamics of super organisms, to function in a way of a single organism.

2. RELATED WORK

Orit Peleg and team to quantify the response of the honey bees developed an experimental setup by mechanical shaking over short and long time intervals (changing the frequency). For the purpose of forming the cluster, they attached a caged queen to a board and allowed the cluster to form around her. The board was moved horizontally and vertically at a frequency ~1Hz. This paper discusses the method of creating optimized visualizations for working on simulations.

3. FORCES UNDER CONSIDERATION

In order to understand how honeybee moves in a free space, we consider honeybee to be a neutral particle in space and when one honeybee corresponds with another honeybee, we consider that to be interactions based on Newton's first order forces.

3.1 Lennard-Jones Force:

Lennard Jones Potential is a mathematically simple model that is used to approximate the interaction between a pair of neutral atoms or molecules. One of the variations of LJ Potential can be given as:

$$u_{LJ}(r_{ij}) = 4\epsilon \left[\left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^{6} \right]$$

Where ε governs the strength of the interaction,σ defines the length scale and r_{ij} defines the distance between two particles. The term of the 12th order dominates when the particles are closer together and models the repulsion. The term with the 6th order dominate when the particles are away from one another and models the attraction.

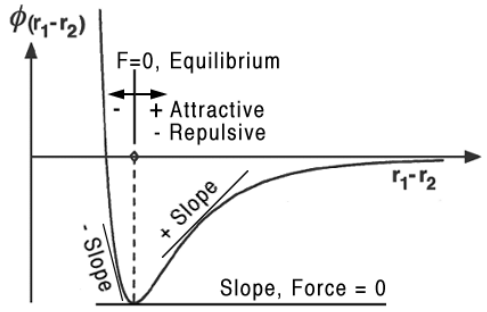


Fig 3.1: Potential well formed using LJ Potential
(source : "Molecular Dynamics: An Introduction" by Lloyd Fosdick)

Due to both forces of attraction and repulsion, there is a possible potential well (drop) at an optimal distance where the force is minimum. It is often said to be a state of equilibrium and

during the state of equilibrium the particles do not move and try to form static triangular lattice. This is often considered as converging point and all the simulations are carried out until point of convergence is reached. The distance between two particles (r_c) where potential energy is minimum is called threshold distance and it is about 1.12 m.

Another important thing to note was that we had developed a thresholding distance as 2.5m above which there will be no force acting on the particles.

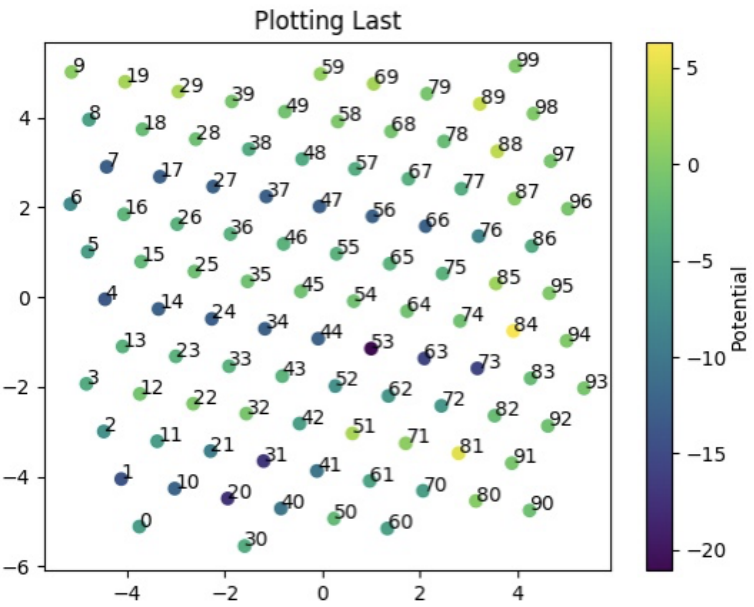


Fig 3.2: Equilibrium state after convergence.

3.2. SPRING FORCE:

According to Hooke's law, the force (F) has to extend or compress the spring by some distance X. As we are working with a real world organism, which is acting in swarm structure, it is fit to consider each particle as spring system with respect to the adjoining particles. The stiffness of the spring system is governed by the spring constant (k).

Spring force is given as follows:

$$\text{Force } F_{spring} = kr_{ij} \text{ if } r_{ij} > r_c \text{ else } 0$$

Here r_{ij} refers to the distance between two spring systems interacting and r_c acts as the cut-off for the systems. In our case, we chose r_c as 1.2 * equilibrium distance.

4. SYSTEM MODEL

The system was implemented in python due to the ease of encapsulating the objects and its ability to render into a backend of the system when converting it into a full stack application. Usually molecular dynamic system is modeled using procedural languages, and incorporating matrices as the data structure.Not following the usual route, we decided to create an object oriented representation to map the interactions and try out the simulations to be rendered in real time.

We used Flask, microservices framework in python that is useful for creating a minimal application from scratch. Using the REST protocols in Flask, we created a visualization model for simulations to be rendered in real time. The architecture of the application is as follows:

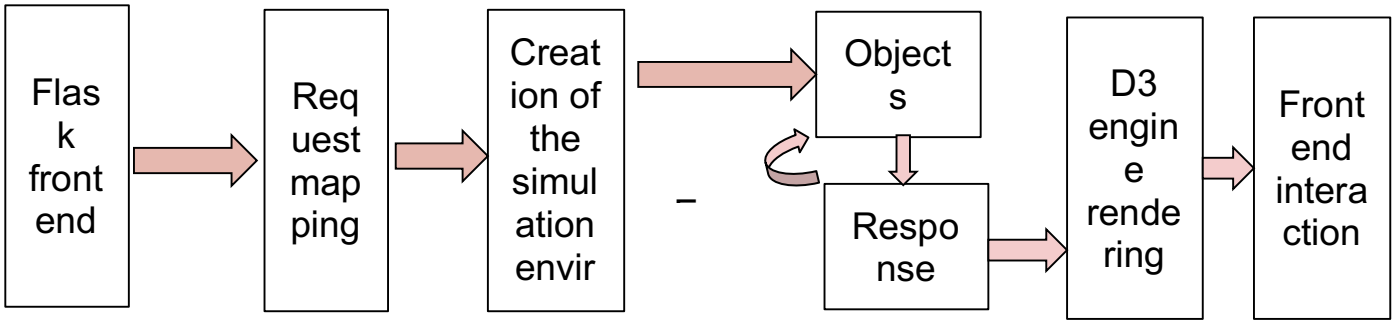


Fig 4.1 : Architecture of the visualization system. The system receives a REST request and based on the request params the system initiates an environment

The lifecycle of the simulation starts with user setting up useful threshold parameters for various functions and creating a request call to the system. The system parses the various parameters and generates a simulation environment for the system to create objects like Particles and interacting forces to start interacting.

At the end of every iteration cycle, based on the printing parameters selected by the user, the response containing the position of the particles are sent back to the d3 visualize engine which is setup during the start. The engine parses out every particle and maps it in the d3 coordinate space.

When plotting the 3d points on the map, the frontend system is capable of rotating the system in a 360°degree of freedom. This type of interaction is independent of the coordinate system, creating a decoupling in implementation.

For horizontal displacement, the particles that are considered to be attached to the surface are moved to a particular position left or right based on the turn. This methodology tries to mimic the horizontal displacement of board in the experiment.

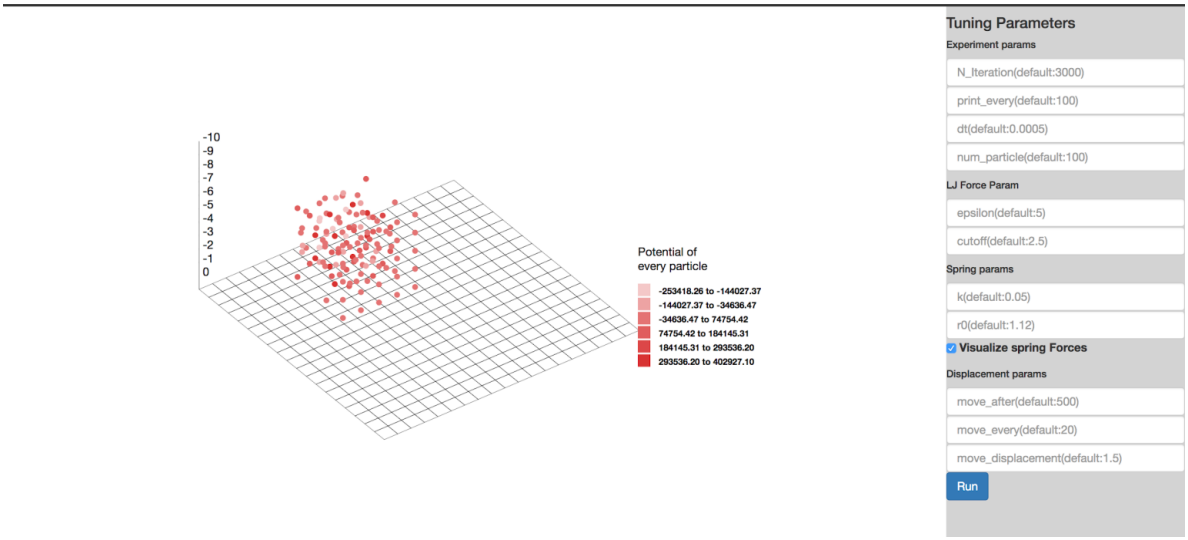


Fig 4:2: Visualization system rendering a 3d model with options to change the tuning parameters on the side. Based on the options

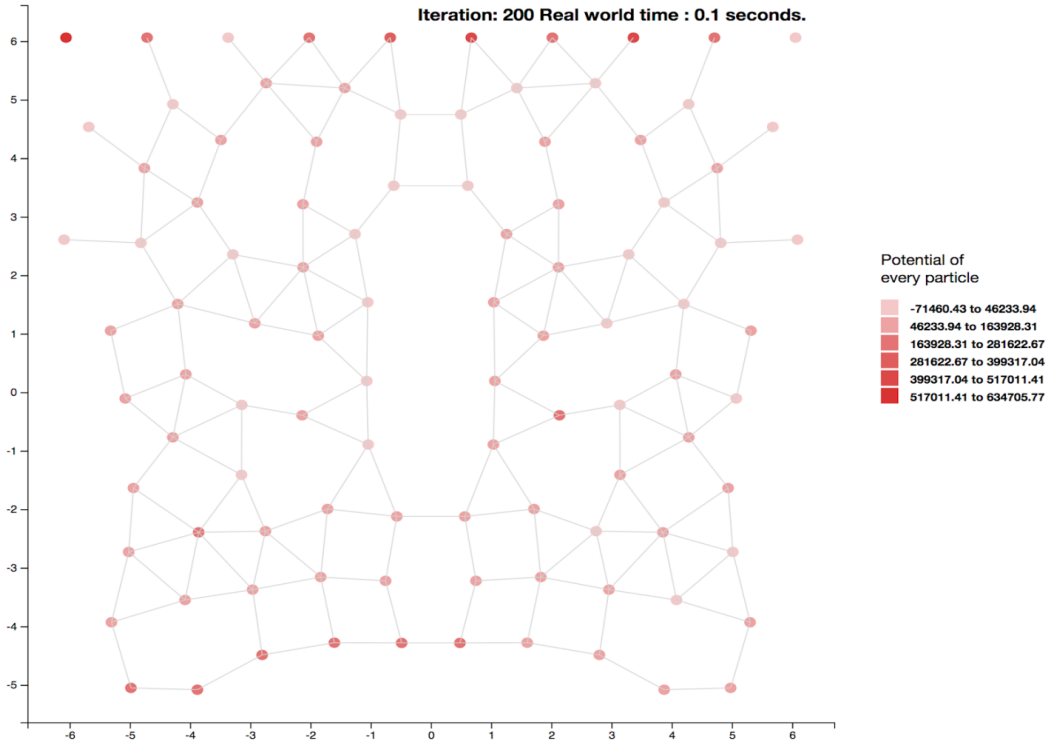


Fig 4.3 : Visualization system rendering a 2d model with spring forces interacting with each other

5. OPTIMIZATION TECHNIQUES

5.1. CELLLIST:

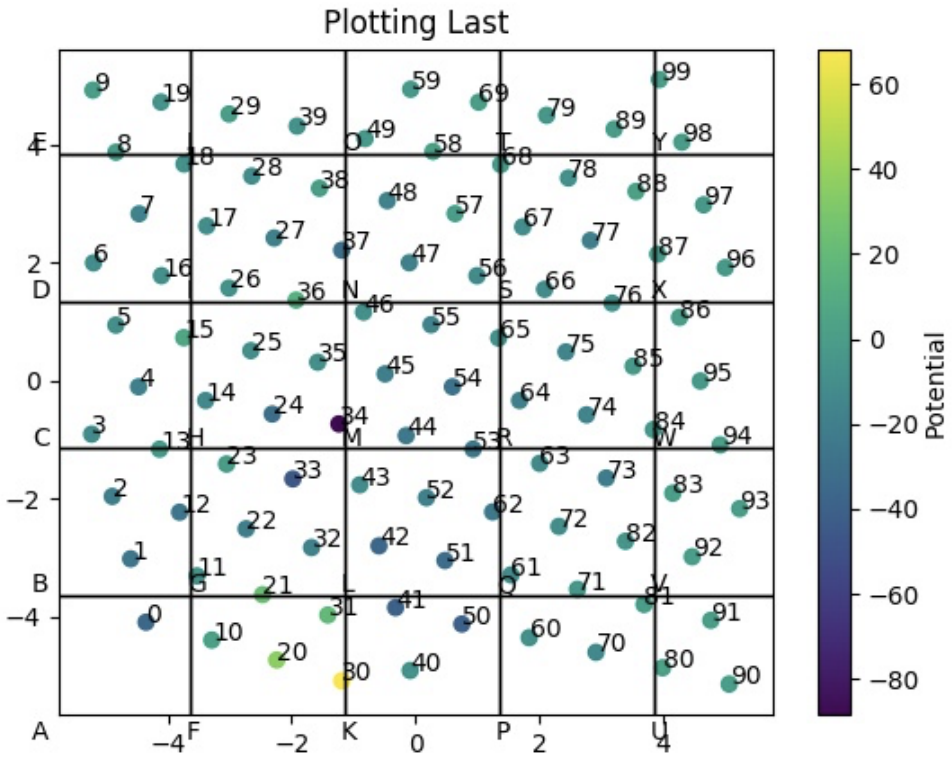
One of the time consuming operations, involve finding the interaction between every other particle without checking the distance in the normal implementation. Any particle that is beyond the threshold distance does not have an interacting force.

Cell lists work by subdividing the simulation domain into cells with an edge length greater than or equal to the cut-off radius of the interaction to be computed. The particles are sorted into these cells and the interactions are computed between particles in the same or neighboring cells.

The cell list algorithm takes advantage of the short-ranged interactions and reduces overall complexity to O(N) by ignoring the majority of non-contributing pairwise interactions.[3]

Algorithm for the scenario would change into:

```
For all neighbouring cells:
      For all particles in cellA:
            For all particles in cellB+cellA:
                  Compute Forces(Particle in cell,Particle in adjcell)
```



Time analysis:

Type of Environment	Number of Particles	Time Taken for 3000 iterations
Without Celllist	100	81.6 seconds (6.91 seconds for graph creation)
With Cell List	100	56.30 seconds (8.04 seconds for graph creation)

Type of Environment	Number of Particles	Time Taken for 500 iterations
Without Celllist	4900	1240.2 seconds (26.91 seconds for graph creation)
With Cell List	100	506.30 seconds (18.04 seconds for graph creation)

5.2. MULTITHREADING:

As python only support pseudo multithreading, we tried implementing threading concept for every particle force calculation in separate thread and tried to join the thread before the next operation. As the weight on every thread was very less, a mere O (1) multiplication it did not do anything better than a single threaded problem.

6.REFERENCES

[1] . Orit Peleg, Jacob M. Peters, Mary K. Salcedo, Lakshminarayanan Mahadevan
bioRxiv 188953; doi: https://doi.org/10.1101/188953

[2] Zhenhua Yao,Min Cheng,Jian-Sheng Wang . New O(N) neighbor list method for molecular dynamics

[3] Dobson, Matthew; Fox, Ian; Saracino, Alexandra . Cell List Algorithms for Nonequilibrium Molecular Dynamics

[4]. Alper Sarikaya and Michael Gleicher. Scatterplots: Tasks, Data, and Designs. IEEE Transactions on Visualization and Computer Graphics

[5] Tutorials on d3 : bl.ocks.org

[6] Tutorials on debugging and Flask programming : http://flask.pocoo.org/