Physically Based Animation
Assignment 1

Yinan Wang 48-176630 Enea de Bollivier 48-179717

Introduction

In this assignment, we implemented an animation system without using any external animation libraries. This work is done in Java using Java Swing for UI. It's important to note that we set particle in 3D but the visualization is in 2D on x and z. We just added this possibility for later 3D graphics implementation.

The first step was to create the class Particle that we would use in all the assignment. Depending on the Task, we modified it a little but it mainly remain unchanged.

Then we had to create a way to view our simulation. We decided to implement a class Viewing that handle everything related to the visualization window. If we want to simulate in 3D, we just have to modify this class and use a 3D library.

All Task are divided in their own directory.
To launch the program. Compile Taskn file and launch it.

Example :
javac Task1.java
java Task1

Task 1

The task 1 required us to modelize a basic action of a force on a particle. We chose to use the Verlet integration in order to avoid any divergence.

The method to calculate the position “calculatePos()” can be found at Particle.java line 52. Initial position, velocity and gravity are given in Task1.java line 12, 13, 15 respectively.

Task 2

Based on Task1, we modified the method for calculating the position, as shown in Particle.java method calculatePos() line 5 We set the window frame as four walls, and check whether there's collision in each iteration. If collision occurs, we push the particle back to border and reverse the velocity. This method conserve the energy.

Task 3

To make the ball stay on the wire, we use the same method as the one used for collision to modify the position. The only difference is that we do not modify the speed as long as it is not hitting and end of the line.

Task 4

The first problem of this task, was to detect collision without an exponential complexity. This is made in the checkCollision () function Collision.java : line 298 . We choose to implement the Bounding Volume Hierarchy to do this. First step is to divide the area into cluster. For this we choose to use the K-mean algorithm (see references). The algorithm is implemented in Collision.java line 73 :

Clustering(int[] tab). We create a Tree of the Clusters. This way, all nodes contain a cluster of at least two particles, all leaves of the tree contain only one particle.

Once we have the tree, we have to check for collision. This is implemented is the RecursionCheck() function Collision.java line 247.

Task 5

For each particle we calculate the force applied by each other particles. The complexity of this is $O(n^2)$.

Task 6

To lower the complexity of the force calculation. We used the clustering method used for checkCollision(). We modified the Clustering program in order to also calculate the position of the mass of a cluster and its mass. Then we use the CalculateForce() in Collision.java line 76 to recursively calculate the force on applied by clusters on a particle.
With this we reach a nlog(N) complexity. For every node in the tree we only do one operation. The Tree width is equal to the number of particles. So the depth is equal to log(N). The complexity is depth*width= nlog(n).

For the same initial conditions, you can see that we obtain the same results as Task5. This proved that our method worked.

Task 7

For the Netwon Cradle, we just tried to reuse our collision methods implemented earlier.

To simulate the limitation of the movement of one ball. We use the method showed in the course for the bead on a wire. We did not manage to reach a full transmission of the energy as some of it remains in the balls in the center. (This is why they move a bit).

Sources :

https://www.datascience.com/blog/k-means-clustering
https://en.wikipedia.org/wiki/K-means_clustering

tutorials for graphics
https://docs.oracle.com/javase/tutorial/uiswing/painting/step1.html