Computer Lab 2 - A pile of particles

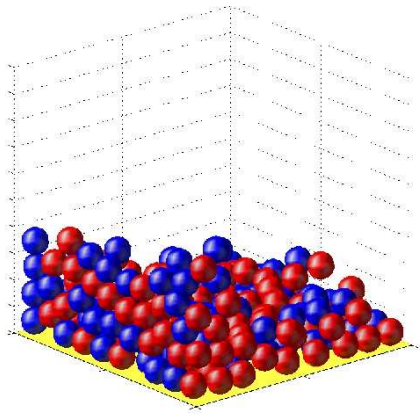

Figure 1: A pile of spherical particles in a box.

1 Introduction

Piling of many objects is a classical test of physics engines. The goal is to pile as many objects as possible with a minimal amount of computational time and without artifacts on the physics, e.g., jittery behavior or large penetrations between the bodies.

The purpose of this computer lab is to simulate a number of spherical particles bouncing on top of each other and understand the basics of modeling and simulation of collisional contacts. As contact model you will apply both the *impulse method* (instantaneous transfer of momenta) and the *penalty method* (linear springs) should be implemented and tested. Make use the theory on particle systems in *Notes on Discrete Mechanics*.

2 The task

Implement both the *impulse method* (instantaneous transfer of momenta) and the *penalty method* (linear springs) for spherical particles. Analyze the preservation of total energy, total momentum and the stability of the methods.

2.1 Basic level - mandatory

- Create a simulation of a system of spherical particles (circular in 2D) that may move along one direction in space – the same direction as that of gravity – and collide with each others and with the ground surface using the impulse method (hardcore particles). Include effects of dissipation in the collision law.
- Create a simulation of a single spherical particle colliding with the ground surface using the penalty method, i.e. soft particles using linear damped spring. It is not necessary to visualize any deformation of particles as in Figure 2 - that is an optional bonus task.
- In both cases, verify that the total energy is preserved when there is no dissipation in the system (restitution $e = 0$ and spring damping $k_d = 0$). Show this by presenting data from the simulation, either in a figure plot or table of showing how the total energy varies over time.
- In both cases, verify that a single particle reaches the same height in between each bounce in the absence of dissipation.
- In the case of the impulse method, verify that the total linear momentum is preserved when two particles collide in the simulation. Try different masses, velocities and with and without dissipation.
- Demonstrate both simulations with snapshots at different time.

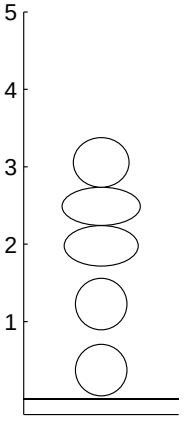

Figure 2: The basic (mandatory) level is to simulate particles moving along a vertical axis.

2.2 Advanced level - optional bonus tasks

Observe that the maximal amount of bonus point on each computer lab is five points. You may do all of the optional tasks, but can maximally earn five bonus points.

- Extend to several soft particles (linear spring) on top of each others. Verify with the simulation that when two particles collide, the total linear momentum is preserved. Try different masses, velocities and with and without dissipation. **3 bonus point**.
- With the impulse method, analyze how many particles that can be placed on top of each other with the given time-step and mass. Are there any artifacts? Can the method be improved to remove these artifacts. **1 bonus point**.
- With the penalty method, analyze how many particles that can be placed on top of each other with the given time-step and masses. When the particles become too compressed, compensate by increasing the spring stiffness. Eventually the simulation becomes unstable and no more particles can be added to it. **1 bonus point**.
- Visualize the soft particle deformation of the particles, e.g., as in Figure 2 **1 bonus point**.
- Let the particles be confined to a 2D box or a 3D box instead of just along a line in 1D. Show snapshots of the simulation. **2 bonus point for each of the impulse and penalty method**.

3 Specifications

Where nothing else is said, use the following parameters

$$
\begin{aligned}
h^{\text{soft}} &= 0.001\ s \\
h^{\text{hard}} &= 0.01\ s \\
g &= 9.81\ m/s^2 \\
m &\sim 1\ kg \\
r &\sim 0.3\ m \\
k &\sim 4000\ N/m
\end{aligned}
$$

4 Hints

- See the sample code mass_spring.m for a MATLAB/Octave example on how to structure the code, data types and simple graphics.
- You may choose to visualize the particles in either 2D or 3D. See the the sample code graphics_2D.m for example in MATLAB/Octave on how to handle graphics in the simulation. The key is to use *handles* for fast update of the figure window. The sample code moving_spheres.m visualizes moving 3D spheres.
- See the the sample code graphics_2D.m for example on how to visualize a deformable circle (advanced optional exercise).