Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

computer graphics laboratory

PHYSICALLY-BASED SIMULATION COURSE 2018

EXERCISE 3 - COLLISION HANDLING

Handout date: 10.10.2018

General Rules

Setup. Please go to our gitlab repository (https://gitlab.vis.ethz.ch/cglphysics/PBS18-Exercises) and carefully follow the instructions to update and run your forked project.

Goal of this exercise

In this exercise, you will handle collisions between rigid bodies with the following ways:

- Impulse-Based Collision Response
- Broad Phase Collision Detection
- Narrow Phase Collision Detection

Problem 1: Impulse-Based Collision Response

Statement. We have multiple rigid bodies where they will collide in several time steps. In this problem, you are required to implement an impulse-based collision response to update the momentum of each object. The magnitude of impulse $\mathbf{j} = j\mathbf{n}$ is derived as follows:

$$j = \frac{-(1+\epsilon)\mathbf{v}_{rel}^{-}}{m_a^{-1} + m_b^{-1} + \mathbf{n} \cdot (\mathbf{I}_a^{-1}(\mathbf{r}_a \times \mathbf{n})) \times \mathbf{r}_a + \mathbf{n} \cdot (\mathbf{I}_b^{-1}(\mathbf{r}_b \times \mathbf{n})) \times \mathbf{r}_b} \tag{1}$$

We recommend you to read David Baraff's course note (https://www.cs.cmu.edu/~baraff/sigcourse/notesd2.pdf) from the page 40 to 47 to follow the derivation of the j. Please fill in `applyImpulse()` method of `CollisionDetection.h` in order to update the linear and angular momentum of object a and b.

Relevant functions. There is an instance `contact` of `struct Contact` in `CollisionDetection.h` which points two colliding objects (already detected by collision detector).

Goal. You should see multiple colliding objects as seen in Fig. 1.

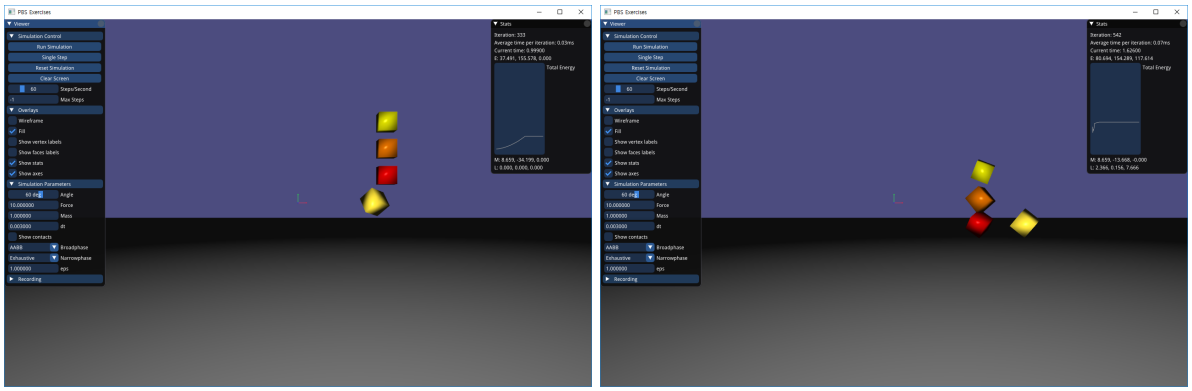

FIGURE 1. Screenshot of an example result of problem 1.

PROBLEM 2: BROAD PHASE COLLISION DETECTION

Statement. In this problem, you can select one of broad phase collision detection algorithms to speed-up the overall collision detection pipeline. As an example, we provide an implementation of Axis Aligned Bounding Box (AABB), and you are required to reduce the amount of time for collision detection in narrow phase by implementing a better broad phase method. We recommend, for instance, Sweep and Prune or one of Space Partitioning algorithms.

Please implement other broad phase collision detection algorithm in `computeBroadPhase()`.

Goal. For now, collision detection with AABB is roughly 10× faster than one without it. Speed up the collision detection pipeline! You can check the average time per iteration on the status window located at the top-right corner.

PROBLEM 3: NARROW PHASE COLLISION DETECTION

Statement. The final step is to improve narrow phase collision detection part. We provide an implementation of exhaustive search algorithm between all pairs of vertex-face and edge-edge combinations. You can select, for instance, Separating Axis Theorem (SAT) which is an improved version of Sweep and Prune or Gilbert-Johnson-Keerthi (GJK) method.

Please implement other narrow phase collision detection algorithm in `computeNarrowPhase()`.

Goal. The same goal as Problem 2.