Dartmouth COSC 89.18/189.02 Computational Methods for Physical Systems, Winter 2019

Instructor: Bo Zhu (bo.zhu@dartmouth.edu)

Assignment 2: Particle Physics

Reading

In this assignment we are implementing the basic collision detection and response algorithms for rigid particles (LV1) as well as an SPH particle fluid simulator (LV2). Before getting started, please read the course notes 2 and 3 and slides for lecture 6 and 7.

Also, please read the paper "Particle-Based Fluid Simulation for Interactive Applications" by Matthias Muller et.al. in 2003 before starting the fluid simulator.

Starter Code

Check out the latest starter code for assignment one from the course GitLab: https://gitlab.com/boolzhu/dartmouth-phys-comp-starter The project for Assignment 2 is in proj/a2_particle_physics. The code can be compiled using CMake (the same way as for Assignment 1).

The main change we made in the starter code for Assignment 2 is that **the simulation is interactive**. We get rid of the file IO part, write the results to memory directly, and visualize everything on screen interactively. You may also interact with the simulation by clicking the left mouse button to add new particles in the simulation.

Requirements

You are expected to implement your own particle physics simulation for this assignment. There are three-level requirements including particle collision and response, particle fluid simulation, and a technical paper implementation, corresponding to 1, 2, and 3 points (remember that you need 4 points for the full 40% assignment credits for the entire quarter).

Level-One (10%):
(Deadline Feb 12)
Implement the algorithms for particle-environment collision detection (2%) and response(3%), and particle-particle collision detection (2%) and response (3%).

Or, you may implement the SPH particle fluid simulator, including the particle-neighbor search (4%) and force updates (6%).

Level-Two (20%):
(Deadline Feb 19)
Implement both parts of LV1 (10% each).

Level-Three (30%):
(Deadline Feb 21)
Implement the major part of one of the classical technical papers in the reading list or any original paper related to SPH fluid simulation (except Muller's 2003 paper) or multi-physics coupling simulation. You may take advantage of the starter code to help your paper implementation.

Implementation Tips

Collison Detection and Response:

The only file you need to modify is ParticleSand.h, where you are expected to implement four functions:

- Particle_Environment_Collision_Detection(),
- Particle_Environment_Collision_Response(),
- Particle_Particle_Collision_Detection(),
- Particle_Particle_Collision_Response().

There are two types of objects in the scene: spheres (stored in particles) and general implicit geometries (stored in env_objects). An implicit geometry class implements two functions: Phi(x) returning the signed distance for a given position, and Normal(x) returning the normal vector for an input position. By default, we have one implicit object (the spherical terrain) stored in the array of env_objects and four spheres stored in particles. After pressing 'p', the four spheres will start falling due to the gravity without any collision handling.

In *Particle_Environment_Collision_Detection*(), you need to detect all collisions between *particles* and *env_objects* by writing a nested for-loop. For a detected collision, the index of the particle and the index of the env_object is stored as a Vector2i in the array of *particle_environment_collision_pairs*.

In *Particle_Environment_Collision_Response*(), you need to process each particle-object pair in the *particle_particle_collision_pairs* by adding a penalty-based collision force to each particle. The force is calculated based on the signed distance and the normal of the particle's current position (see the course notes).

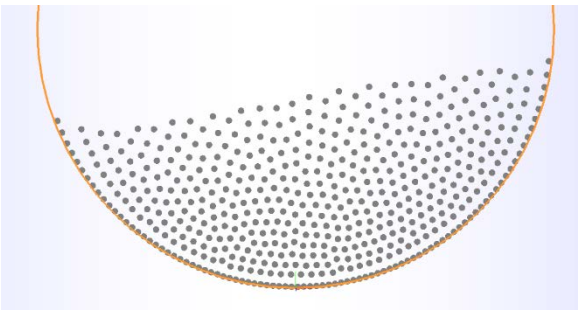
In *Particle_Particle_Collision_Detection*(), you need to detect all collisions among *particles* by writing a nested for-loop (we are not using acceleration structures here). For a detected collision, the index of the two particles are stored as a Vector2i in the array of *particle_particle_collision_pairs*.

In *Particle_Particle_Collision_Response*(), you need to process each pair of colliding particles by adding penalty-based collision forces to each of the two particles (see the course notes for calculating the force).

Once you finish the four functions properly, you should be able to run the simulation of particle collisions by adding new spheres into the scene interactively. Press 'p' to start/stop the simulation.

SPH Fluid:

The only file you need to modify is ParticleFluid.h, where you are expected to implement the neighbor search function in the SpatialHashing class and a number of density and force calculation functions in the ParticleFluid class. The guidance for implementing each function can be seen in the code comments and the particle fluid course notes.



Once you finish the implementation, you should be able to run the simulation of SPH fluids of a bulk of particles falling onto a spherical container. Press 'p' to start/stop the simulation, and left click the mouse to add new fluid particles.

Submission

For Level-One and Level-Two assignments, you need to submit your source code (including ParticleSand.h, or ParticleFluid.h, or both, and any other pieces of code you created for the assignment), a README file for any specific issue, and a video for your simulation.

For Level-Three assignment, you need to submit your source code package, your data and executable, and schedule a time with the instructor and the TA to present your implementation briefly.