Advanced Computer Graphics
Exercise 5 - Particle System Part 1

Handout date: 26.10.2015

Submission deadline: Monday, 2.11.2015, 11:00pm

Note

Undeclared copying of code or images (either from other students or from external sources) is strictly prohibited! Any violation of this rule will lead to expulsion from the class. No late submission allowed.

What to Hand In

Solve the exercise in groups of 3. Hand in a compressed zip file renamed to `Exercise5-firstname_lastname-firstname2_lastname2-firstname3_lastname3.zip` that contains the following files:

- `Mass_spring_viewer.cpp`
- A readme file (.txt or .pdf) containing the following: a description of your solution, how much time you needed, encountered problems (if any), and answers to the questions.

New Framework

This exercise introduces a new framework to run particle simulations. This is the first part of a two part exercise and therefore there's additional functionality in the framework that will only be used next week. As in all previous exercises, you are given a QtCreator project file that allows you to build and run the system.

There are several keyboard inputs that control the simulation:

- Once the system is running you can use the number keys to change between different predefined scenes. For this exercise we only work with scenes `1`, `2`, and `3` (the remaining scenes will be used in part two of the exercise next week).
- `t` and `T` let you increase and decrease the time step of the simulation.
- Pressing space starts/pauses the simulation.
- With `v` you can toggle visualizing the particle forces.
- `f` allows to change between different types of external forces (details below).

You are encouraged to experiment and create new scenes yourself. All your implementation tasks are in `Mass_spring_viewer.cpp`.

5.1 Euler Integration (14 points)

Implement the `Euler` case in the method `time_integration`. You have to update all the particles' positions, forces, and velocities. For the computation of the forces, use the `compute_forces` method.

5.2 Center Force (4 points)

All the subsequent exercise parts compute different forces that act on the particles. These forces must be implemented within the `compute_forces` method. The parts that you need to complete are annotated with 'to do' tags.

To get started, you are asked to implement a simple (physically implausible) center force that pulls each particle back towards the origin. The magnitude of the force is proportional to the distance from the origin: $\mathbf{F}_c = c\,(\mathbf{0} - \mathbf{p})$, where $\mathbf{0}$ is the origin and $\mathbf{p}$ is the position of the particle.

If you have Euler integration and the center force working, run scene `1`. If you choose the coefficient $c = 20$, you should see a particle oscillating around the origin and until the simulation becomes unstable. You'll need to press `f` to activate the center force, by default no external force is activated.

5.3 Damping (4 points)

Implement viscous damping $\mathbf{F}_d = -\gamma\mathbf{v}$, where $\mathbf{v}$ is the particle's velocity. The damping coefficient γ is given as member `damping_` of class `Mass_spring_viewer`. Damping keeps the simulation more stable.

5.4 Gravity (4 points)

Implement gravitation force $F_g = 9.81m$. You can activate the gravitation force by pressing `f` twice.

5.5 Boundary Collisions (14 points)

In the code we provide four planes (in normal form) that define a rectangular boundary for the simulation. You have to implement force based point-plane collisions. The penalty force that you apply is proportional to the penetration distance d and along the plane normal $\mathbf{n}$: $\mathbf{F}_k = k_{coll} d\mathbf{n}$. The coefficient is given as member `collision_stiffness_`. To compute the penetration distance, you will also need the radius of particles, which is given as member `particle_radius_`. You can verify your implementation by using scene `2` and by activating gravity.

5.6 Mouse Interaction (10 points)

This part enables the user to interact with the simulation by pulling around individual particles with the mouse. Clicking on a particle and dragging will apply a force on the particle. We model that mouse force as a damped spring between the selected particle and the mouse cursor. You find the equation for that force in the course slides. Note, you can assume the rest length of the mouse spring to be 0 and the cursor velocity to be 0 too. Both positions are provided, and the spring stiffness and damping coefficients are also given as members of `Mass_spring_viewer`.

5.7 Spring Forces (10 points)

Implement damped spring forces. You should already have most of the code form the mouse interaction. All additional information required for this task is given in the class `Spring`. Inside `Mass_spring_viewer`, you can access the vector of all springs in the scene with `body_.springs`. You can test your implementation with scene `3`.