OpenGL - Lab Session 6
Physical Animation

TODO: introduce animate method, was removed from TP4

The purpose of this lab session is to run a physically-based animation, using a particles-springs model.

1 Integration Scheme

The aim is to compute the position x of a particle. According to the Taylor series (2nd order) :

$$\begin{aligned} x'(t+h) &= x'(t) + hx''(t) \\ x(t+h) &= x(t) + hx'(t) + \frac{h^2}{2}x''(t) \end{aligned}$$

where h is the integration time step, x' (first derivative of x) the velocity of the particle, and x'' (second derivative of x) its acceleration. This scheme is exact for constant accelerations (e.g. free fall).

The acceleration x'' must first be computed thanks to classical mechanics (Newton, ...). An integration scheme (explicit/implicit, Euler, Runge-Kutta, ...) then enables to compute x' and x.

Newton's second law The acceleration of a particle i of stricly positive constant mass m_i on which each particle j exerts the force (or action) $f_{j\to i}$ is given by :

$$x_i'' = \frac{1}{m_i}\sum_{i\neq j} f_{j\to i}$$

Newton's third law To every action there is an equal and opposite reaction i.e. for all (i, j) :

$$f_{i\to j} = -f_{j\to i}$$

2 Force models

Gravity On Earth, gravity g is uniform and exerts a force, called weight, proportional to the mass (note that g is a vector and not a scalar) :

$$f_{G\to i} = m_i g$$

Springs Springs are used to simulate an elastic behavior that tends to bring two particles back to a given distance from each other, the *equilibrium length*. An ideal spring is parametrized by its equilibrium length l_0 and its stiffness k. The force exerted on particle i is then :

$$f^k_{j\to i} = -k\left(l_0 - ||x_i - x_j||\right)\frac{x_i - x_j}{||x_i - x_j||}$$

Damped springs Ideal springs quickly yields to unstable simulation. To prevent this, *damping* a spring is used to reduce the relative motion between two particles. The action of a damper is proportional to the viscous damping coefficient k_v and the relative velocity (difference of velocities of the two particles projected on the spring direction) :

$$f^{k_v}_{j\to i} = -k_v\left((x_i' - x_j')\cdot\frac{x_i - x_j}{||x_i - x_j||}\right)\frac{x_i - x_j}{||x_i - x_j||}$$

The total action of a damped spring is computed by summing the different contributions of an ideal spring and a damper :

$$f_{j\to i} = f^k_{j\to i} + f^{k_v}_{j\to i}$$

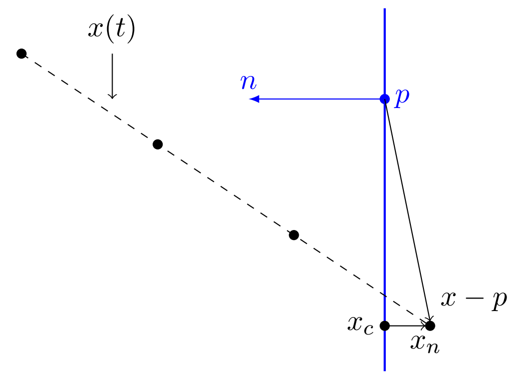
Viscosity of the medium The movements of all bodies moving within a medium are damped according to its global viscosity. This phenomenum can modeled by an action proportional to the medium viscosity coefficient v and exerted on the opposite direction of the velocity :

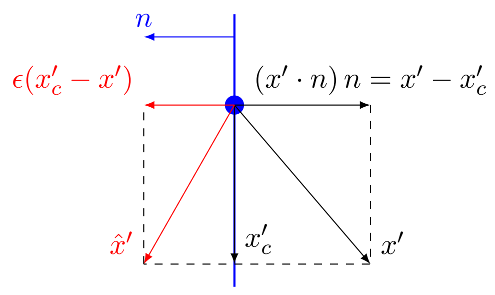
$$f_{V \to i} = -v x'_i$$

3 Collisions

The next step is to study the collisions between a particle x and a plane, represented by a point p and a normal n. Processing a collision requires three steps :

1. **Collison detection :** While detecting collisions is a very complex problem in the general case, it is quite simple between a particle and a plane. A particle penetrates the plane if the distance from the particle to the plane is negative i.e. :
$$(x - p) \cdot n < 0$$
2. **Impact :** Once the collision is detected, we compute the configuration the objects would be in if the impact was completely absorbed. In our case, we want to compute the position x_c of the particle if the plane was completely absorbing the impact energy. This is approximated by projecting the particle and its velocity on the plane, i.e., with notation $x_n = ((x - p) \cdot n)\, n$:
$$\begin{aligned} x_c &= x - x_n \\ x'_c &= x' - \big(x' \cdot n\big)\, n \end{aligned}$$
3. **Rebound :** The rebound corresponds to the restitution of the energy absorbed during the impact. It is represented by the coefficient ϵ (parameter `rebound` in the lab session) usually with values between 0 (completely absorbant impact) and 1 (completely elastic impact). The difference between the actual impact and the completely absorbant impact is taken into account proportionally to ϵ :
$$\hat{x}' = x'_c + \epsilon(x'_c - x')$$
i.e. :
$$\hat{x}' = x' - (1 + \epsilon)\, \big(x' \cdot n\big)\, n$$





4 Building your own particle-spring system

4.1 Provided Code

The provided code enables to model a system composed of particles linked together with damped springs, within a medium with gravity and viscosity. Particles a represented by small spheres, with a radius and a mass. The initial scene is composed of a static particle that can be controlled by the mouse, a dynamic particle and a fixed plane. Collisions are handled to ensure non-penetration of the plane by the system particles. The physics code is incomplete, and you will need to write part of it.

This code relies on the previous QGLViewer-based practicals. New elements are introduced :

Dynamics classes two basic classes `Particle` and `Spring`, and a main class `DynamicSystem` (which extends `Renderable`) holding the scene and the physics resolution.

Integration in the viewer Controls were added to manipulate a `DynamicSystem` object with the mouse (`Ctrl`+click to move one of the particles) or keys (toggle gravity, viscosity, collisions detection, ...).

4.2 Integration scheme

Compile and execute the provided code (press enter to start the animation, and the Home button to restart the animation). A particle remains fixed while the other translates.

Question 1 :

Analyse the `DynamicSystem` class. What kind of integration scheme is used ?

Gravity is taken into account (it can be activated/desactivated using the `g` key). However, the velocity of the particles remains constant. To correct this, complete the integration scheme in the method `animation()` by updating the velocity of each particle depending on its acceleration. Observe the results.

Question 2 :

The fixed particle is a dynamic object, just like the moving one. Why is it fixed then ?

4.3 The particle-spring model

You are now going to progressively build an articulated chain of spheres as a particle-spring system. This means that the particles will be linked two by two by damped springs.

Question 3 :

In the method `createSystemScene()`, add a spring between the fixed particle and the moving one. To do so, create a new `Spring` with the parameters defined in `DynamicSystem`, and add it to the `springs` collection.

The scene can be interactively animated using the mouse. When the `Ctrl` key is pressed along with the middle or right button, mouse movements actually translates the fixed particle of the scene. Let's play!

Question 4 :

Model an articulated chain hanging from the fixed ball at one extremity. Choose the same mass and radius for all particles and the same parameters for all springs. Add any number of particles to form a chain and observe the results. Modify the different parameters and observe how the system reacts to each modification.

Question 5 :

To dissipate the velocities, add a global viscosity to the medium to apply a damping force to all particles. Visually, what is the difference between this global damping and the spring damping? (viscosity of the medium can be activated/desactivated using the `v` key)

To better observe the dissipation phenomenon, increase the initial velocity of the particles and observe the animation with and without damping or rotate the mobile particle around the fixed one with and without damping. To better observe the influence of the spring damping, vary the damping coefficient of the springs.

4.4 The collision model

You are now going to handle collisions for a more realistic animation (collisions can be activated/desactivated using the `c` key).

Question 6 :

Uncomment the necessary lines to treat collisions between the particles and a fixed ground plane. Analyse the function `collisionParticleGround` to understand the process of collision management. Experiment with your articulated chain.

Question 7 :

Inspiring yourself from the particle-plane case, handle the collisions between two particles in the method `collisionParticleParticle`.

If two particles p_1 and p_2 don't wheight the same, the reaction can be pondered by $\frac{m_1}{m_1+m_2}$ and $\frac{m_2}{m_1+m_2}$ so that the heaviest object moves less.