MoSIG1 - ICG 08/04/2009

OpenGL - Lab Session 6

Physical animation - Notions

This short lesson briefly presents the notions of physical animation needed to complete this lab session.

1 Integration scheme

We want to compute the position q of a mass. According to the Taylor series (2nd order) :

$$\dot{q}(t+h) = \dot{q}(t) + h\ddot{q}(t)$$

$$q(t+h) = q(t) + h\dot{q}(t) + h^2\ddot{q}(t)$$

where h is the integration time step, $\dot{q}$ the first derivative of q (i.e. the velocity of the mass) and $\ddot{q}$ is the second derivative of q (i.e. the acceleration of the mass). This scheme is exact for constant accelerations (e.g. free fall).

Thus, we have to compute the acceleration, i.e. the second derivative of the system ($\ddot{q}$). This accelerations will be computed thanks to classical mechanics (Newton, …). This will result in realistic animations and shapes.

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 :

$$\ddot{q}_i = \frac{1}{m_i}\sum_{i\neq j} f_{j\to i}$$

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

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

2 Action 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) :

$$f_{G->i} = m_i g.$$

Spring : Springs are used to simulate an elastic behavior that tends to bring two particles back to a given distance from each other. This distance is called equilibrium length. The spring action depends on its equilibrium length l and on its *stiffness* k :

$$f^k_{j\to i} = k(\|q_i - q_j\| - l)\frac{q_i - q_j}{\|q_i - q_j\|}.$$

Damped spring : Damping a spring is used to reduce the relative motion between two particles. The action of a damper is proportional to the viscosity ν and the relative velocity (difference of velocities of the two particles projected on the spring direction) :

$$f^{\nu}_{j\to i} = \nu((\dot{q}_i - \dot{q}_j).\frac{q_i - q_j}{\|q_i - q_j\|})\frac{q_i - q_j}{\|q_i - q_j\|}.$$

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

$$f_{j\to i} = f^{k}_{j\to i} + f^{\nu}_{j\to i}.$$

Many natural phenomena can be represented by damped springs, for instance deformable bodies.

Damping : The viscosity of the medium damps the movement of the bodies moving in it. This phenomenum is modeled by an action proportional to the viscosity coefficient ν and exerted on the opposite direction to the velocity :

$$f_{V\to i} = -\nu\dot{q}_i.$$

3 Collisions

We are now going to study the collision between a particle and a plane. Processing the collision is done in three steps :

1. **Collison detection :** The collision detection is a very complex problem in the general case. For a particle and a plane, the particle penetrates the plane if the distance from the particle to the plane is negative i.e. :

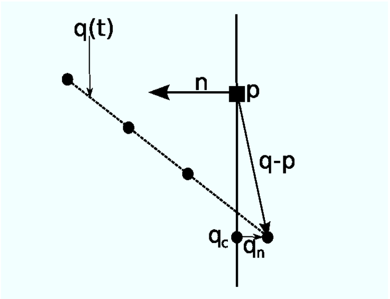
   $$(q - p).n < 0$$

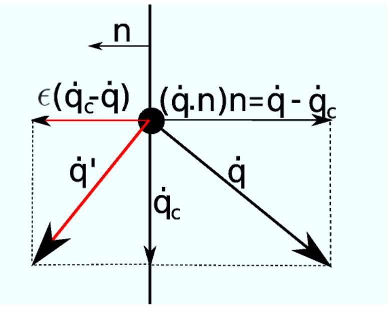
   where the plane is represented by a point p and a normal n.

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 q_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 $q_n = ((q - p).n)n$:

   $$q_c = q - q_n$$

   $$\dot{q}_c = \dot{q} - (\dot{q}.n)n$$




n
$\epsilon(\dot{q}_c-\dot{q})$
$(\dot{q}.n)n=\dot{q}-\dot{q}_c$
$\dot{q}'$
$\dot{q}_c$
$\dot{q}$

3. **Rebound :** The rebound corresponds to the restitution of the energy absorbed during the impact. It is represented by the coefficient ϵ (variable `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 :

   $$\dot{q}' = \dot{q}_c + \epsilon(\dot{q}_c - \dot{q})$$

   i.e. :

   $$\dot{q}' = \dot{q} - (1 + \epsilon)(\dot{q}.n)n$$

OpenGL - Lab Session 6

Physical animation - Practice

In this lab session, we are going to implement two physical animation notions : mass-spring systems and collisions. The sheet on notions of physical animation will help you understand the provided code and answer the questions. Read the first 2 chapters of this sheet.

1 Code

Compile and execute the provided code : a ball translates and a ball is fixed.

Provided code : First analyse the file *tp6.cpp* : there are the previously seen methods (camera, time control, display, . . .) and some new methods :

- `setDynamics()` that initializes an object of class `Dynamics` that implements physical animation models ;
- `drawBalls()` and `drawSprings()` that display an articuled chain ;
- `mouseClick()` and `mouseMotion()` that manage the actions controlled by the mouse. Indeed, the position of the fixed ball is controlled by the mouse.

Initial scene : The initial scene is composed of :

- a fixed plane ;
- a fixed ball controlled by the mouse ;
- a ball with an initial velocity.

The animation is controlled by the object `dynamics` of class `Dynamics`. At each time step, the animation function of `dynamics` is called (cf : `updateAnimationParameters()`). This function computes the new position of the masses. Then, the balls representing the masses as well as the springs will be drawn by `drawBalls()` and `drawSprings()`.

2 Integration scheme

Question 1 : Analyse the files `dynamics.h` and `dynamics.c`. Specifically, analyse the computation of the forces and the integration scheme used.
The gravity is taken into account. However, the velocity of the ball remains constant. To correct this, complete the integration scheme by updating the velocity depending on the accelerations (stored in `forces`). Observe the results.

Question 2 : The fixed ball is a dynamic object just like the moving ball. Why is it fixed then?

3 Mass-spring model

We are now going to progressively build an articulated chain of balls as a mass-spring system. This means that the balls will be linked two by two by a damped spring.

Question 3 : In the method `setDynamics()`, add a spring between the fixed ball and the moving ball with the method `addSpring()` of `Dynamics`. Use the declared variables for the spring parameters. Observe the results on the animation by moving the fixed ball with the mouse.

Question 4 : Model an articulated chain hanging from the fixed ball at one extremity. Choose the same mass and radius for all balls and the same parameters for all springs. Parameterize the number of created balls with the variable `nbBalls`. Observe the results.
Modify the different parameters and observe how the system reacts to each modification.

Question 5 : To dissipate the velocities, model a damping force applied on each ball. Visually, what's the difference between the damping we've just introduced and the spring damping?

Tip : To better observe the dissipation phenomenon, increase the initial velocity of the balls and observe the animation with and without damping or rotate the mobile ball around the fixed ball with and without damping. To better observe the influence of the spring damping, vary the damping coefficient of the springs.

4 Collision model

We are now going to add collisions for a more realistic animation. Read the corresponding chapter in the lesson.

Question 6 : Uncomment the necessary lines to treat collisions between balls and planes. Analyse the function `collisionBallPlane()` to understand the process of collision management. Experiment with your articulated chain.

Question 7 : Inspiring yourself from the ball-plane collision, implement the ball-ball collision in `collisionBallBall()`.