Project 1: Particle systems

Due date: 22.05.2014

Overview

In this project you will implement a *particle system* with *constraints*. You must implement at least the required features, see below; you will also have to make a movie (demo) showing your system in action. The class will vote for the best demos, and the top two winners will receive extra credit. Additionally, you may implement some of the listed extensions (or invent your own !) for extra credit.

Skeleton Code

You have been provided with some *skeleton code* which you may use to jump-start your coding. Basically, the only thing the code does is move three particles around randomly, and draw some (nominal) constraints and spring between them. Further, the code does little more than implement basic window management and graphics.

Required Features

Your code must implement the following features:

- **Generalized force structure.** This is described in the slides. (If you use the skeleton code, you should replace `delete_this_dummy_spring` by a `std::vector` of forces.) Then, you'll use your generalized force structure to implement two forces:
  - `GravityForce.` Acts like gravity...
  - `SpringForce.` A damped spring between two particles. (Skeleton code is already provided.)
- **Generalized constraint structure.** This is also described in the slides. (If you use the skeleton code, replace `delete_this_dummy_rod` and `delete_this_wire` by a `std::vector` of forces.) You must implement at least the following two subclasses:
  - `RodConstraint.` Constraints two particles (x_1, y_1) and (x_2, y_2) to be a fixed distance r apart, i.e.,

$$C(x_1, y_1, x_2, y_2) = (x_1 - x_2)^2 + (y_1 - y_2)^2 - r^2. \tag{1}$$

    (Rendering code included in the skeleton.)

  - `CircularWireConstraint.` Constraints a particle (x, y) to be at a fixed distance r from some point (x_c, y_c), i.e.,

$$C(x, y) = (x - x_c)^2 + (y - y_c)^2 - r^2. \tag{2}$$

- **Mouse interaction.** When the user clicks and drags the mouse, a spring force should be applied between the mouse position and the given particle to make your system interactive.

- **Several numerical integration schemes.** The integration scheme should be selectable at run-time with keystrokes or GLUT menus. You will find this easiest if you implement a pluggable integration architecture as described in the slides. The minimum integration schemes are:

  - Euler
  - Mid-point
  - Runge-Kutta 4.

  Experiment with each of these and make a comparison w.r.t. to breaking point when increasing the time step.

- **2D Cloth.** Create a rectangular network of particles with appropriate springs holding it together. For example, you can imagine a curtain which either slides along a line (holding it in place) or has two (or more) fixed points. Sliding or fixing points should be regareded as constraints and implemented using the method of Lagrange mutipliers. Further, experiment with spring configurations and constraints, and report your findings (which worked/didn't work, why ?).

- **Simple interaction with the 2D cloth.** Based on the above constraints and mouse interaction. Implement a simple interaction mechanism, by which you can apply a horizontal force to the cloth. Collisions of the cloth with a line (wall) should also be experimented with.

- **Angular Springs.** Pulls a triplet of particles so that their subtending angle approaches some rest angle. Based on these, build a simple hair(-like) simulation.

Your demo must be able to turn each of these features on and off individually, so that they can be verified.

Remark: **Groups of three** people should implement **implicit integration** as described in the class. Note that, code for solving linear systems by the Conjugate Gradient method is provided within the skeleton code.

Optional Features

- **Verlet Integrator**. See http://en.wikipedia.org/wiki/Verlet_integration.

- **Leapfrog Integrator**. See http://www.physics.drexel.edu/students/courses/Comp_Phys/Integrators/leapfrog/.

- **Collisions with other particles**. Particles bounce off each other.
- **Angular Constraints**. Similar to angular springs, but the angle is actually constrained.
- **3D**. Implement and render the simulations in 3D.
- **Visualization**. Implement simple visualizations which allow you to visually inspect forces, velocities, etc. of your particles.
- **Implicit integration**. See above remark about three-people groups.
- **Hair with collisions**. How can we implement this ?

Deliverables

- **Code.** On the due date, you must deliver a CD containing the code of your project. Instructions about how to compile and run your project should also be included.
- **Movie.** Should be delivered on the CD. You use the movie in the demo day, to show (off) your results. The skeleton code contains functionality to save snapshots from your code. The snapshots can be coalesced into a movie using several packages (ffmpeg, VirtualDub, ImageMagic, etc.).
- **Fast-forward presentation.** You give this short presentation (2–3 minutes) in front of the class, in the demo day. Use it to tell the audience what you did and point to the methods you used.
- **Paper.** The paper (minimum 4 – 5 pages) describes the results you obtained, and the methods you used (implemented). Snapshots, code fragments, equations, timings, etc. should all be included. The paper should be delivered as a hard copy to the instructor.