

CSE 872: Computer Graphics

Homework Assignment #1

Due date: Sept 29th 2011 – 11:59pm

Platform required: Visual C++ / Windows.

Submission procedure: A zip file of the Visual C++ project by email to ytong@msu.edu

Assignment: Implementation of a 2D particle system.

Details:

You need to complete all the following points, and provide your implementation and executable that must run under Windows (along with special libraries if needed). Don't lose your time on a sophisticated interface, keys are just fine (provide a README file to explain the important keys).

The goal of this assignment is to implement a simple 2D particle system, with tunable Lennard-Jones forces as inter-particle forces.

- A. Write a procedure animate. Input: positions and velocities of the particles, and a time step dt. Output: update positions and velocities after integration over dt. Use explicit Euler integration.
- B. Define the inter-particle forces as:
$$F_{\text{internal}}(\text{exerted on } \mathbf{p}_1 \text{ by } \mathbf{p}_2) = \lambda \left(\left(\frac{r_0}{r}\right)^{2a} - \left(\frac{r_0}{r}\right)^a \right) \frac{(\mathbf{p}_2 - \mathbf{p}_1)}{r} \quad (\text{with } a=4)$$
$$F_{\text{friction}}(\text{exerted on } \mathbf{p}_1 \text{ by } \mathbf{p}_2) = \mu(r) \|\mathbf{v}_2 - \mathbf{v}_1\| (\mathbf{v}_2 - \mathbf{v}_1)$$
with
$$\mu(r) = \begin{cases} -1/(2r_0^2) r^2 + 1 & \text{if } (r < r_0) \\ 1/(2r_0^2) (r - r_0)^2 - 1/r_0 (r - r_0) + 1/2 & \text{if } (r > r_0) \text{ and } (r < 2r_0) \\ 0 & \text{if } (r > 2r_0) \end{cases}$$
where \mathbf{p} is a 2D position, $r = \|\mathbf{p}_2 - \mathbf{p}_1\|$, \mathbf{v} a velocity, λ and r_0 are real numbers, and a is an integer. Notice that you must define $F_{\text{internal}}=0$ for particles that are more than $2r_0$ away.
- C. Define a 2D scene with at least a floor (horizontal line), and a square on the floor as an obstacle for the particle
- D. Implement a collision detection unit (floor: test if $y < y_{\text{floor}}$; square: test if (x,y) lies in the square)
- E. Write a procedure that handles collisions. Input: old positions and velocities (at $t-dt$) and new positions and velocities (at t). Output: new, relevant positions so that no particles are below the floor or inside an obstacle. (Hint: you can just project particles to the closest outside point, or something similar). THINK!
- F. Add an interaction with the mouse (one particle is attracted to the mouse cursor using a stiff spring)
- G. Start your demo with the particles placed on a regular grid (3x3 for starters)
- H. Use a better data structure to make your code work in linear time (instead of quadratic). Try more particles.
- I. Let the user play with the different parameters of the force.

Questions ? email me.