15-464/15-664 MiniProject 2

Due **March 18** at start of class

The goal of this project is to experiment with simulation. The six project options below give you choices to develop your own simulator or experiment with existing ones depending on your interests.

I would like to see a nice implementation of one of these options and that you at least make a serious attempt at a second one. If you do only one project, I expect a detailed investigation of the topic that may go well beyond the descriptions below.

You may do your own work or work together in teams of two.

You may also choose your own project. You must get my approval first. The scope should be similar to the projects below.

Option 1. Spring-Mass Cloth simulation.

Objective: create spring mass cloth and investigate its behavior.

You will create cloth that consists of mass nodes connected by springs and dampers. Explore the effect of different stiffness and damping settings on the following:

- Simulation speed (how large can you make your timestep?)
- Simulation stability
- Cloth appearance

Can you create multiple instances of cloth that appear to have very different behaviors?

Extensions (not required):

- Handle contact with the environment
- Handle self-collisions
- Explore controlled cloth, where you can use "fake" control forces to coerce the cloth into different shapes.

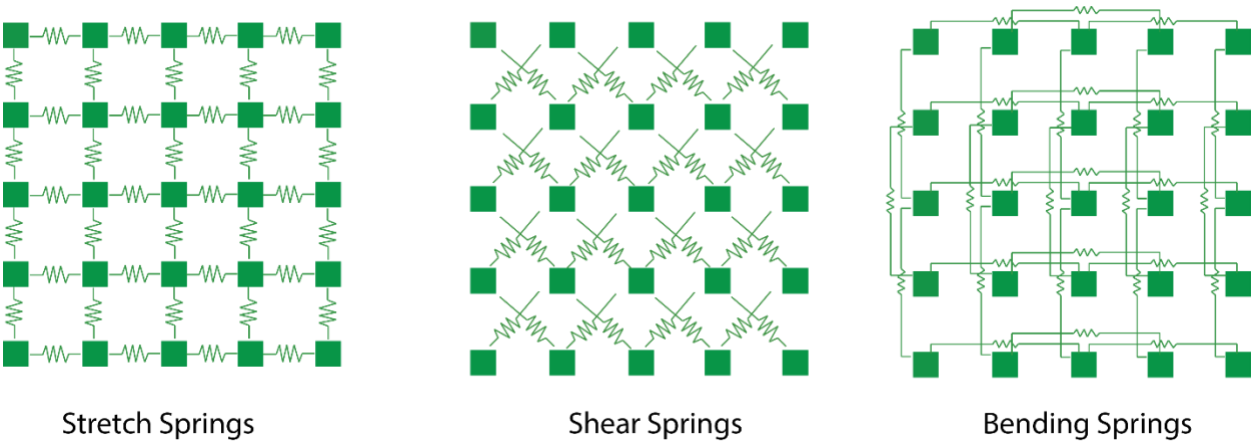
Implementation:

You may write your own simulator for this project or use an existing simulator. You could also start with the modified AMCViewer here: http://graphics.cs.cmu.edu/nsp/course/15464-s20/www/assignments/AMCViewer-simulation.zip. For AMCViewer, you will need to implement functions in Cloth.cpp, Particle.cpp and SpringForce.cpp.

Mass-Spring Model

The Mass-Spring Model is a very basic method to simulate cloth. It is relatively easy to implement, and with some care can produce good results, so it is a good exercise before exploring more advanced methods.

In the Mass-Spring Model, cloth is simulated as a grid of particles, which are connected to each other by spring-dampers. Each spring-damper connects two particles and generates a force based on the particles' positions and velocities. We will use three types of spring dampers that are shown below: ***stretch springs*** *(also known as structural springs)*, ***shear springs***, and ***bend springs***. In addition to these spring forces, each particle is also influenced by gravity. The layout of this model is illustrated in the figure below:



Stretch Springs | Shear Springs | Bending Springs

Computing Forces

Gravity: gravity is simply a force proportional to the mass of the particle. We can use the equation:

$$f_g = mg$$

Spring-Dampers: A spring-damper is defined by three constants: its spring constant k_s , a damping factor k_d, and its rest length l_0. We can compute the force using the equation:

$$f_1 = -\left[k_s(\|p_1 - p_2\| - l_0) + k_d\left(\frac{(v_1 - v_2)\cdot(p_1 - p_2)}{\|p_1 - p_2\|}\right)\right]\cdot\frac{p_1 - p_2}{\|p_1 - p_2\|}$$

$$f_2 = -f_1$$

Resources:

[1] https://steven.codes/blog/cloth-simulation/

Option 2. Constraint based Cloth simulation.

Objective: create constraint based cloth and investigate its behavior

You will create cloth that consists of mass nodes connected by length constraints. You may write your own simulator for this project or use an existing simulator. You could also start with the modified AMCViewer described in Option 1.

Questions to investigate:

- Can you make your constraint based cloth unstable?
- How does number of constraint update iterations affect cloth behavior?
- Can you create multiple instances of cloth that appear to have very different behaviors?

Extensions (not required):

- Handle contact with the environment
- Handle self-collisions
- Explore controlled cloth, where you can use "fake" control forces to coerce the cloth into different shapes.

Resources:

Ken Toh did a final project for this class some years ago, which turned into his Masters Thesis. Details of his implementation may be helpful to read, and are available in this document: http://www.cs.cmu.edu/~ytoh/stickyfingers.pdf

Some videos from the resulting system are here: http://www.kentoh.com/publications/

Option 3. Experiment with different integrators

Objective: understand the effect of different integration techniques on simulation speed and stability

Create a simulation environment for testing. It can be as simple as the single spring mass system we examined in class, it can be your cloth system from Option 1, or any other environment of your choice that has some interesting dynamic behavior.

Implement three (or more) different integrators. Compare:

- Simulation speed (how large can you make your timestep?)
- Simulation stability
- Simulation accuracy (plot and compare the motion of some interesting points in your scene, is it the same for each integration technique and choice of timestep?)

Some choices are:

- Euler integration
- Implicit Euler (try to work out the equations for a simple system)
- RK4
- Semi-implicit Euler
- Verlet
- Leapfrog

Option 4. Experiment with contact models

Objective: understand the effect of different contact models on contact behavior.

Implement at least two different contact models. You may use an existing simulator or write your own. Some options for contact models include:

- Penalty method. This paper provides a clear discussion of the penalty method in Section III and describes how to add friction (not required): Yamane K, Nakamura Y. Stable penalty-based model of frictional contacts. In Robotics and Automation, 2006 [Paper]

- Constraint based method. Use a constraint based model which simply resets colliding points back to the surface and iteratively adjusts the simulation bodies to satisfy constraints.

- Impulse based collision response using a coefficient of restitution. Details for how to implement this option for particles and rigid bodies can be found in Baraff and Witkin's course notes: http://www.cs.cmu.edu/~baraff/sigcourse/

- Solve the Linear Complementarity Problem in a multi-contact situation. Baraff and Witkin's notes give a description of the setup of this problem, and off the shelf solvers exist. This option is ambitious.

Provide some simple test scenes that allow you to demonstrate differences with videos and plots to compare your results.

If you are interested in further reading, this paper gives a very appealing solution for handling large numbers of contacts, but is out of scope for MiniProject2:
http://www.cs.ubc.ca/labs/sensorimotor/projects/sp_sigasia08/

Option 5. Create a scene that tells a story with simulation

Objective: use simulation in a creative manner

Create a scene that uses simulation to tell a story. The scope is not limited to cloth. You will probably want to use an existing simulator, such as

- Bullet plugin for Maya
- Maya nCloth, nHair or nParticle (under "FX" menu):
- NVIDIA Flex plugin for Unity: https://developer.nvidia.com/flex
- MuJoCo: http://www.mujoco.org/
- Dart: https://dartsim.github.io/index.html). This is a research project for simulating robots

Option 6. Understand pros and cons of an existing simulator

Objective: understand an existing simulation engine

Download and install an existing simulation engine. Try to read about how it handles the aspects of simulation that we talked about in class:

- Contact and collision
- Integration
- Can it do deformable bodies?
- Does it use maximal or minimal coordinates for bodies connected by joints?
- What other features seem important / interesting?

Run your simulator through some basic tests. Choose at least three different tests. Here are some suggestions:

- Can you make a block slide down an inclined plane?
- Can you make a sphere roll down an inclined plane?
- What happens when you stack a bunch of blocks in a tower?
- What happens when you drop a bunch of blocks on top of each other to form a stack?
- What happens when you initialize the system with objects colliding with one another?
- Can you make a simple spring mass system go unstable? How easily?
- How fast is a simulation with many simple bodies (e.g., many blocks)?

Create a scene to demonstrate this simulator.

Deliverables and Handing in

We will use Autolab (https://autolab.andrew.cmu.edu/) for handins. Please upload a single compressed file with the following content:

You should provide:

- A README file describing briefly what you did and what you have uploaded into this directory.
- Your code and documentation of how to run it.
- Any source files required to run your code.
- Videos of animations demonstrating your system
- A 10-minute presentation: Submit your slides or website to describe your goals, techniques you used, as well as your results.

If you run into quota or authentication problems, you may email the material to nsp@cs.cmu.edu. Please try to test out autolab handin, however, before the deadline.

If you did not present MiniProject1, you will be asked to present your results from MiniProject2 in class on the due date.