Assignment 4: Simulation and Keyframing

In this assignment, you will implement two techniques for computer animation: physics-based simulation of mass-spring systems, and keyframe animation of articulated characters. Use your code from Assignment 2 to display the computed animation. There are three required tasks and three optional tasks. For full marks, you are required to do all three required tasks **and** at least one of the three optional tasks.

1. Implement a mass-spring system for simulating a single sheet of cloth, as discussed in class. To do this, you will create a collection of particles in a grid layout, connected to nearby particles with springs. Also add a gravity force acting on all the particles, and allow some particles to be labeled as "fixed" in space so that they never move. Perform time stepping using the semi-implicit Euler scheme described in class.

    - The spring constant of shear springs should be less than that of the structural springs, and that of bending springs should be much less. For each type of spring, the damping constant should be much less than the spring constant.
    - Your implementation should be generic enough that you are able to simulate sheets of different sizes (physical length and width), different resolutions (number of particles), and different material properties (mass density and stiffness).
    - Demonstrate your implementation by simulating a 1m × 1m sheet of cloth, initially horizontal with two adjacent corners fixed, swinging down and then back and forth under gravity. Tune the simulation parameters so that the motion looks somewhat realistic, although it may be a bit stretchy.
    - **Optional task:** Replace the structural springs with inextensible constraints, and use position-based dynamics to simulate them. Since the shear and bending springs are much weaker, you can still evaluate their contribution explicitly as before. Only use constraint projections for the structural edges.
        - You will need to do multiple iterations of the PBD loop to ensure that all constraints are close to satisfied. Tune the iteration count to get a cloth that does not significantly stretch.
        - Refer to the original paper by Müller et al. (2006) for more details.
        - Demonstrate your implementation using the same cloth from the previous task, showing that the cloth appears nearly inextensible when using constraints.

2. Modify your simulation to handle collisions of particles with moving obstacles. You may use discrete collision detection, i.e. you only need to check for collisions at the end of the time step. Also, you only need to test the particles of the mass-spring system; you are not

required to handle the case where an edge or face of the cloth is intersecting the obstacle but all particles are outside.

- Your implementation should support obstacles of different shapes, specifically: spheres, finite cylinders, and infinite planes. They may further have a time-varying transformation applied to them. For now, assume a constant linear and angular velocities specified by the user.
- For collision response, include normal forces with user-specified ε, frictional forces with user-specified μ, and position correction to move particles out of the obstacle.
- Include the obstacles as well in the displayed scene. You may wish to draw the spheres and cylinders slightly smaller (or equivalently, check collisions with a slightly expanded shape) so that penetrating edges and faces are not too noticeable. For infinite planes, just draw a large finite square.
- Demonstrate your implementation on a scene with a ground plane, one or more spheres, and one or more boxes. At least one of the shapes should be moving, and the cloth should collide with all of them.
- **Optional task:** Right now, your cloth will happily pass through itself and get tangled, because we are not testing for self-collisions. Fully robust self-collision handling is very hard, but we can do a reasonable job just by handling inter-particle collisions: make sure the distance between all non-adjacent particles is at least Δx, the rest length of the structural edges. This way a particle is unlikely to sneak through the gap between adjacent particles.
  - To speed up inter-particle collision detection, use a uniform grid of size Δx (see Lecture 19). Since we only care about particles at distance Δx or less, you only need to check the adjacent cells for each particle.
  - Treat the distance between nearby particles as a constraint, and apply a projection impulse to both their positions and velocities in the usual way.
  - Set up a simulation where significant self-collisions occur, e.g. when an initially vertical sheet of cloth drops on the ground, and show results with and without self-collisions.

3. Design an articulated character composed of several bones connected with hinge joints. Attach a collider shape to each bone so that the cloth can collide with the character. Animate the character's motion by keyframing the hinge angles.

   - Each bone may be specified by a pointer to its parent bone, the position (in the parent bone's coordinate system) where it is attached, and the rotation axis of the joint. It should also have one or more colliders (spheres / cylinders) whose transformation is defined by the bone's transformation.
   - The character does not have to be a full body, but it should have at least 4 bones,

and a hierarchy of depth at least 2 (e.g. chest → upper arm → forearm) so you can demonstrate that transformations at different levels are being composed correctly. For example, you could model half a character (e.g. chest + 2 arms + head, or hip + 2 legs).

- Design the character's motion by manually specifying hinge angles at various keyframes, and interpolate them using Catmull-Rom splines. Note that when the keyframes are not uniformly spaced, the derivative at a keyframe t_i should be computed as a *weighted* average $\frac{\Delta t_+}{\Delta t_- + \Delta t_+}\left(\frac{\Delta y_-}{\Delta t_-}\right) + \frac{\Delta t_-}{\Delta t_- + \Delta t_+}\left(\frac{\Delta y_+}{\Delta t_+}\right)$, where $\Delta t_- = \Delta t_i - \Delta t_{i-1}$ and so on, instead of the simpler formula given in class.
- Show the animated motion of your character using at least 5 keyframes. Then add a simulated cloth to the scene and show the results as well.
- **Optional task:** Implement inverse kinematics, so that your character can be animated by directly specifying the positions of the end effectors.
    - Assume that, as before, the root bone is fixed and only the hinge angles need to be computed. Note that if you want to be able to achieve an arbitrary 3D position, you will need at least 3 joints with different axes between the root and the end effector.
    - Refer to Bill Baxter's slides which explain several methods to solve the IK problem. Choose any one and implement it.
    - Demonstrate your implementation by showing an example of a motion that would be hard to achieve by manually specifying joint angles, e.g. a character's hand moving in a perfect circle.