Physikalische Modellierung und Simulation

https://graphics.tu-bs.de/teaching/ws1819/PBM

Assignment 4

Present your solutions for this sheet in the exercise on Thursday, December 06, 2018.
In this exercise, you will simulate a solid body that collides with the surrounding geometry.

4.1 Collision detection and handling (33.5 points)

In main.cpp, a solid body is created as a mass-spring system. This allows us to use a very simple form of collision detection: instead of computing collisions between objects of arbitrary shape, we only have to check for collisions of particles with some geometric primitives. For now, we will restrict ourselves to collisions with infinite planes (or, more exactly, half-spaces).

A plane (or a half-space) is defined by its normal vector, \( \vec{n} \), and its (signed) distance from the origin, \( d \), in the direction of the normal vector. For some point \( \vec{x} \), we can easily check if it lies “above” or “below” the plane (with respect to the direction of the normal vector): if \( \vec{x} \cdot \vec{n} < d \), it is below and a collision occurs.

When a collision occurs, a force has to push the particle back to the allowed half-space. This force is proportional to the distance of the particle to the plane, and it acts in the direction of the normal of the plane. The proportionality constant (the “elasticity” of the plane) can be different for each plane.

Draw a small sketch showing the origin of the coordinate system, a plane with its offset \( d \) and its normal vector \( \vec{n} \), and the position of a particle that has just collided with the plane (such that \( \vec{x} \cdot \vec{n} < d \)). Use your sketch to find out how the above-mentioned force can be computed. Then, fill in the function `computeReflectionForce()` in Plane.cpp.

4.2 Friction (33.3 points)

When you start the simulation, you will see that the movement of the solid body still seems quite slippery. To change this, we have to apply a friction force. The magnitude of this force is proportional to the force \( F \) that is pushing the particle onto the plane. This \( F \) is simply the component of the total force acting on the particle that is perpendicular to the plane, or \( F = \vec{F}_{\text{total}} \cdot \vec{n} \). Its direction is always opposite to the particle velocity (because friction always slows a particle down). The proportionality constant (the “roughness” of the plane) can be different for each plane.

It may again be helpful to draw a small sketch of the plane, its normal vector, the velocity and the force on the particle in order to understand what is going on. Then, fill in the function `computeFrictionForce()` in Plane.cpp.

4.3 Simulate a table cloth (33.3 points)

You have now simulated a solid body that collides with its environment. You can use the same technique to simulate a table cloth that lies on a table. First, you will have to derive a new class from Obstacle that describes a plane with finite extent, i.e., the table. You can also just modify your code for a Plane.
Then, in `main.cpp`, change the mass-spring system so that it only consists of a single layer of cloth. It may help to look at the variables `dimx`, `dimy` and `dimz`. Finally, drop your cloth on the table and see if you can make it form wrinkles.