CS 184: Computer Graphics and Imaging, Spring 2017

Project 4: Cloth Simulator

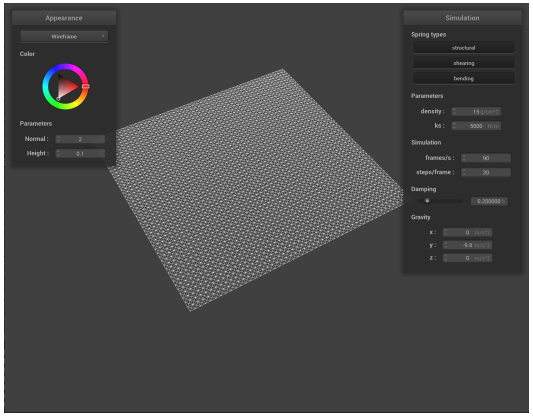
Vincent Tantra, CS184-aed

Overview

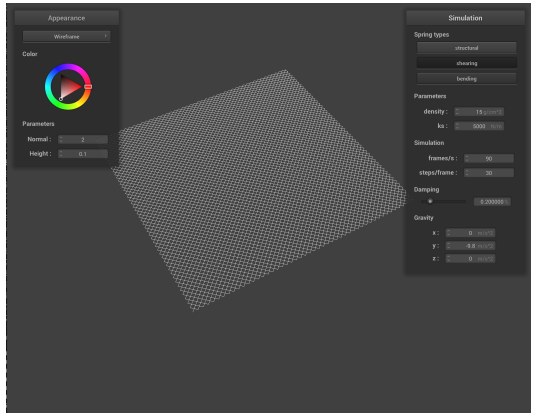
For this project, we create a real-time simulation of cloth physics using a mass-spring based system. I build the structures needed to represent the cloth and define the constraints in between these structures that help control its movement over time. I finally implement final touches such as collisions.

Part I: Masses and springs

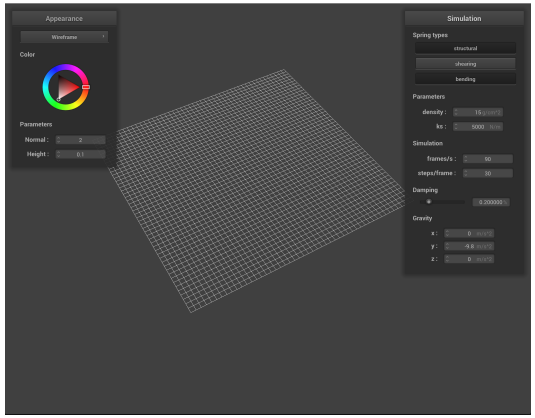
One of the simplest models of cloth graphics is the mass-and-spring model. Simply put, a cloth can be modeled by a large number of point masses that add up to the total mass of the cloth (sort of like a point cloud). These masses have "springs" connecting each other; they have constraints that, if enabled, apply forces between masses so that there is general interaction of motion between cloth particles.
The first step to build this model is to create the grid of masses and springs. We do this by simple iteration,
creating point masses one-by-one in a grid, and then connecting those masses to their respective pairs with springs after all masses have been created. Attached are some images of the resulting wireframe of the cloth. In this case, all constraints are on.



Mesh with all constraints on.



Only shearing on.



Only shearing off.

Part II: Simulation via numerical integration

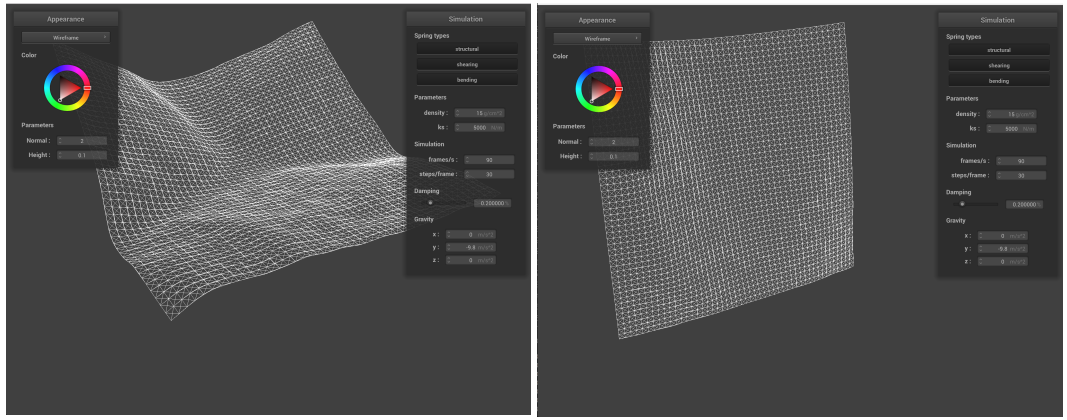
This simulation run by taking timesteps of length dt as time goes on. When a time step happens, we must consider several forces and integrate upon them to find out how our cloth moves. Each point mass is affected by both "external" forces, which are universal forces such as gravity, and "internal" forces, which are the forces applied by the springs that connect them in order to hold the cloth together.

First, we apply the external forces on all point masses. Then, we go through the point masses and calculate how their connected springs act upon these point masses. We accumulate this into a single force vector per point mass and integrate on that.

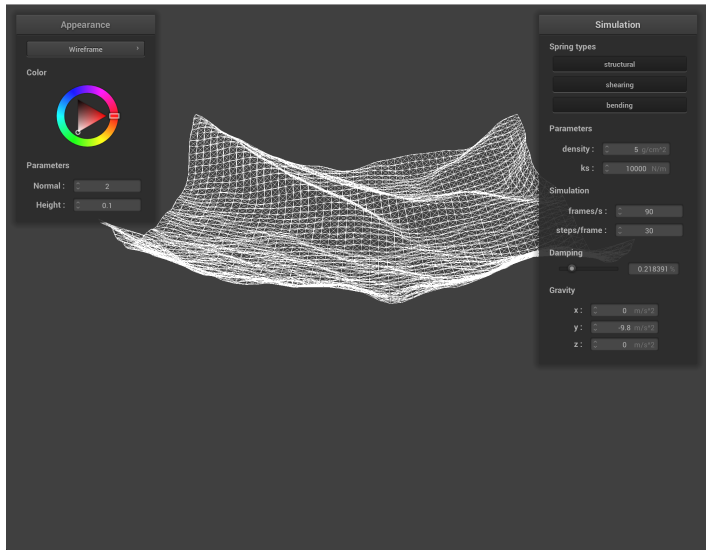
As for how we integrate, we use the Verlet algorithm since it is relatively simple. We take a standard kinematics equation, representing velocity by subtracting a point mass' last position from its current position. We also apply a "damping," a small reduction constant on the velocity as a simple way of representing movement loss due to opposing forces such as friction or heat loss.

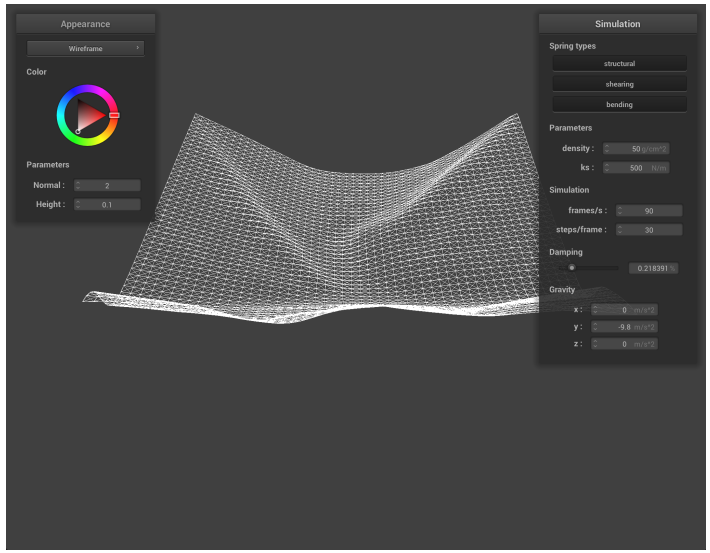
We have one more step before the integration is complete. Some springs become very deformed due to an unnatural combination of forces. The SIGGGRAPH 1995 Provot paper on deformation constraints is the basis for how we correct our model. Fundamentally, we apply a correction to the affected point mass' position so that the spring's length is at most 10% greater than its rest length. We apply half this correction to each point mass, or the full correction to a single point mass if the other one it is connected to is pinned.

This gives us a working simulation. Attached are some screenshots of me playing with the sim.



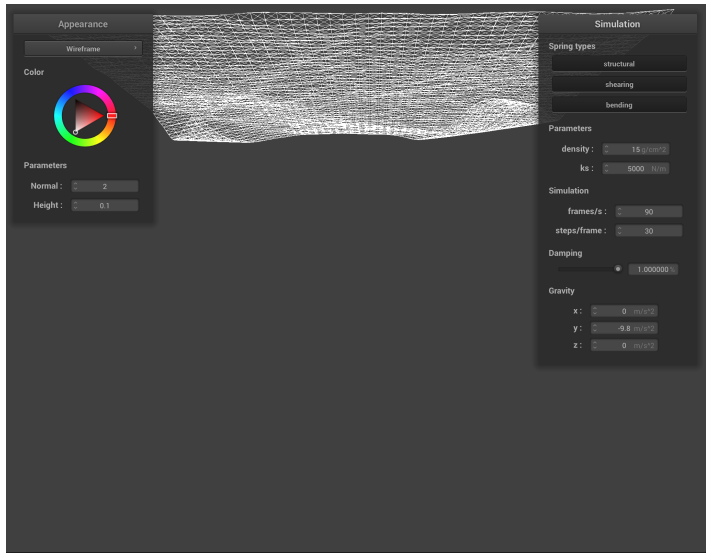
I made a few observations while changing the constants in the simulation. Firstly, the "ks" constant: when it is very high, the forces between the point masses become larger and more "important," so the cloth displays a lot of internal movement; point masses interact with each other at greater levels and the cloth displays a lot of "scrunching," folding up on itself dynamically. When it is low, the greatest force acting becomes the external force of gravity, so the cloth becomes relatively flat and simply falls.
When altering density, the same effect occurs. Extremely low densities exhibit high intra-cloth behavior, such as how high "ks" did, and high densities exhibit more flat behavior.
The damping effect represents the general forces that serve to stop the cloth from moving, the opposing forces. Therefore, when I decreased it, the cloth was in motion for a longer time, with much of its kinetic energy remaining even as it fell to the bottom position, so it continued to bounce and move. If I increased damping, the cloth's movements came to a stop much faster.



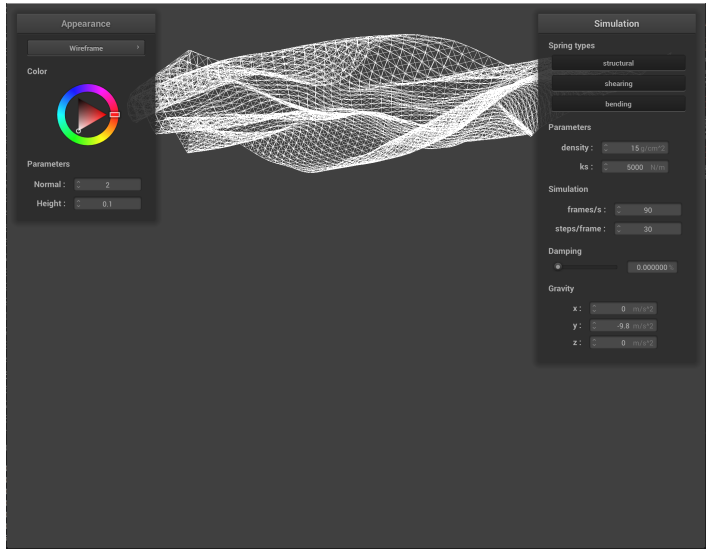


Higher density/lower Ks. Falls a bit more flat.

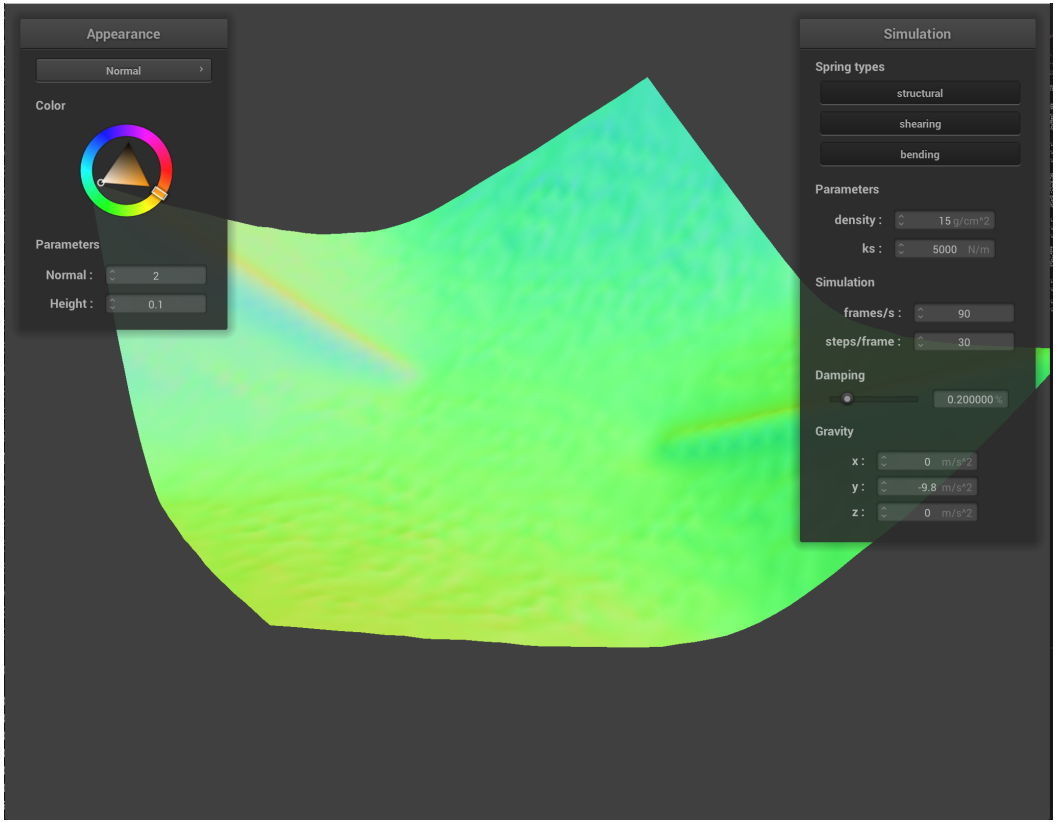
Lower density/higher Ks. The cloth is more dynamic when falling.



High damping. The intra-cloth motion dies relatively early.



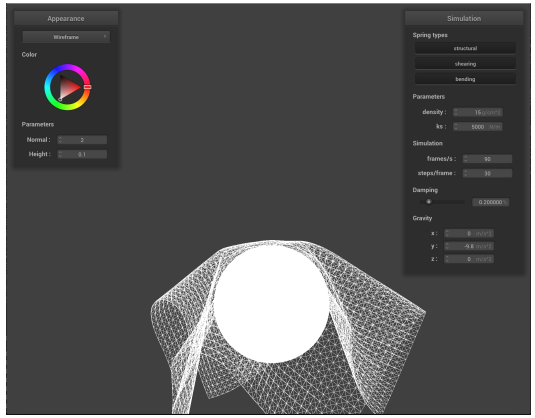
Low damping. Intra-cloth motion is retained and carries on for a while.



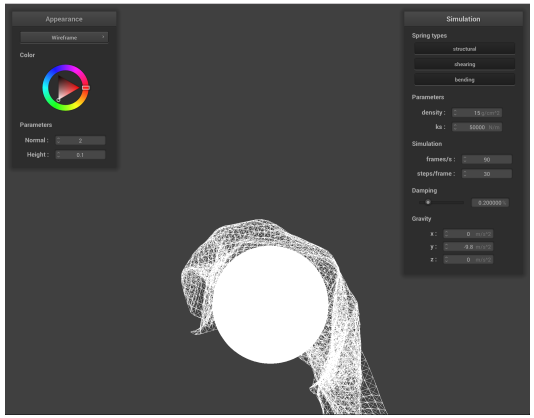
4-pin resting state with shading.

Part III: Handling collisions with other objects

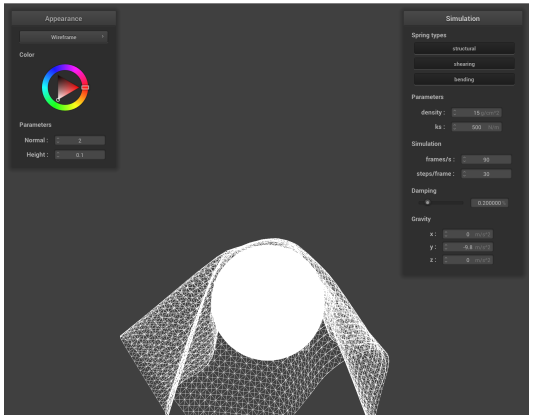
For this part, I check each point mass to see if it either lies inside a sphere or if it would have crossed a plane in that time step, and "bump" it to the proper location on the surface of the object to model collisions. Attached are a few screenshots of how the cloth looks colliding with a sphere. If I change the Ks constant, higher Ks leads to a longer time before the cloth comes to rest on the sphere. In fact, a Ks of 50000 never comes to rest, and continues to interact with itself to the point of falling off the sphere.



Ks = 5000. Falls typically with a bit of motion.

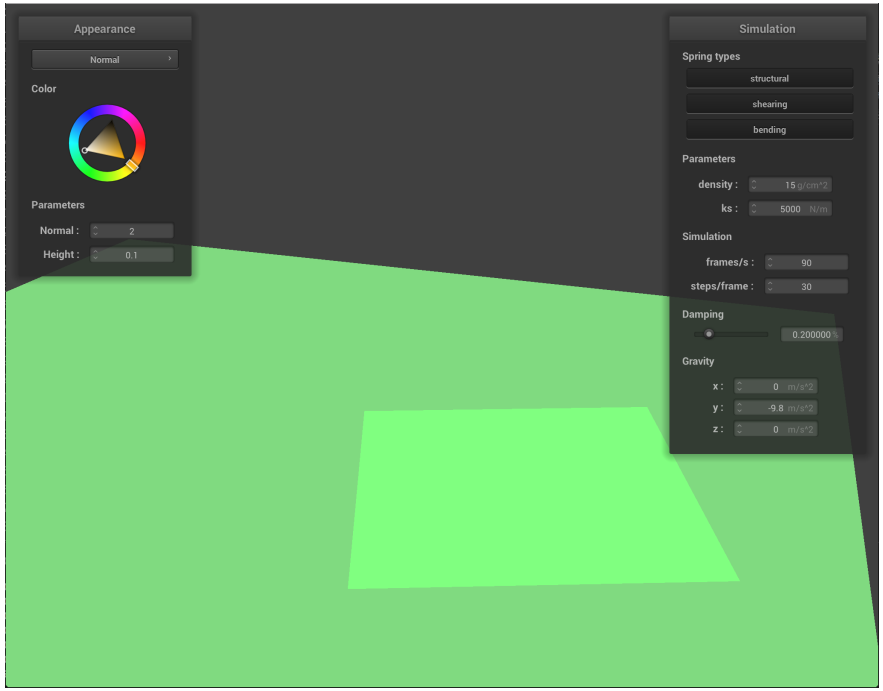


Falls and goes crazy! Lots of motion and never fully stops.



Ks = 500. Falls with more flatness, motion dies faster. Hugs the sphere closer.

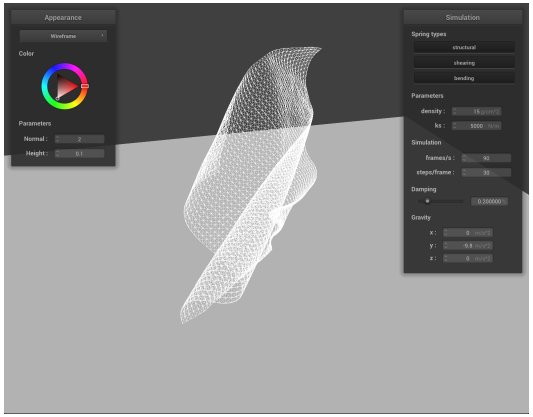
We also implemented support for collision with planes.

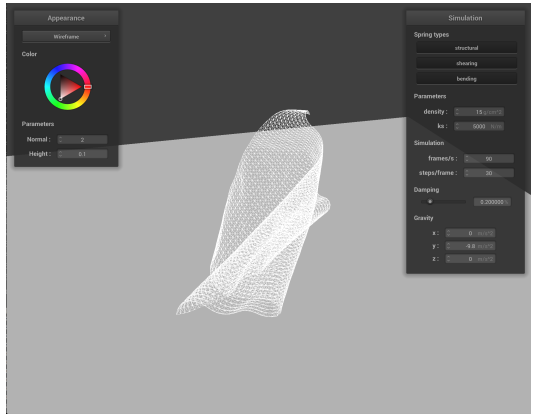


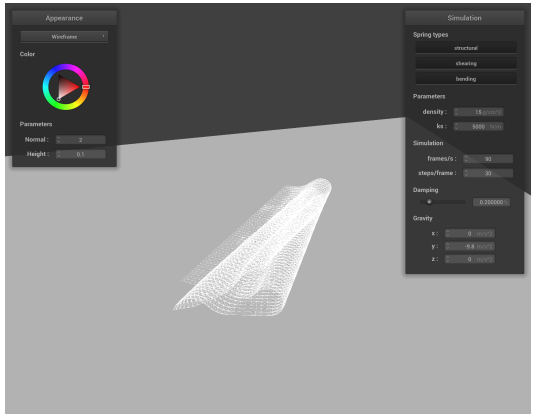
The cloth resting on the plane.

Part IV: Handling self-collisions

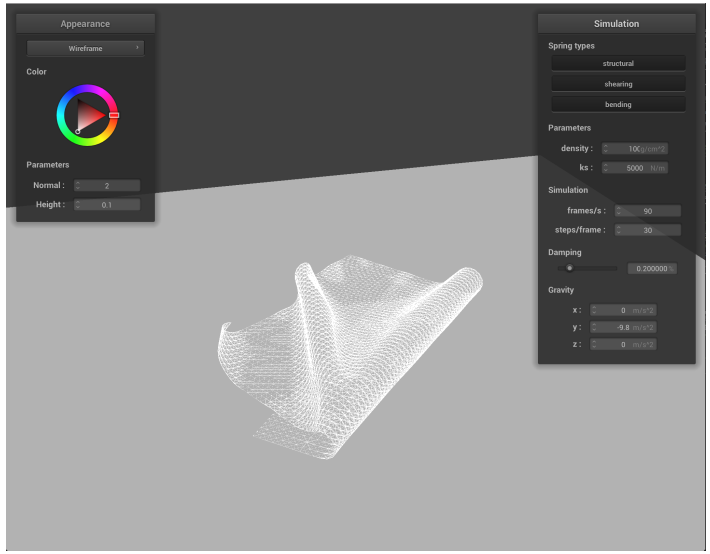
There is one more collision to consider - the cloth colliding with itself. In some video games or animations, this edge case can be seen, where a cloth clips through itself since self-collision isn't accounted for/detected. To implement realistic folding and crumpling of the cloth as it falls upon itself, we need to implement this case as well.
Typically, this is done by detecting if 2 point masses are a certain distance away from each other, and if they are too close, we apply a correction factor just as we have with point masses that are inside spheres or crossing planes. However, it would take too long to check every point mass and see if they are close to every other point mass, so we cut the runtime by hashing the pointmasses based on a unique float that attributes them to a certain "hashbox." Basically, a cloud of pointmasses that belong to the relatively same space, or "hashbox," will all be mapped to that hashbox and have the same unique float. In the hashmap, each float is connected to a vector of pointmasses that are in that box. We can then test collisions for a pointmass by comparing it against all the other pointmasses in its box, effectively scaling down the number of comparisons.
This is what the cloth looks like as it falls vertically upon a plane, during an early stage, mid state, and late stage (rest) of it folding on itself.


Early Stage

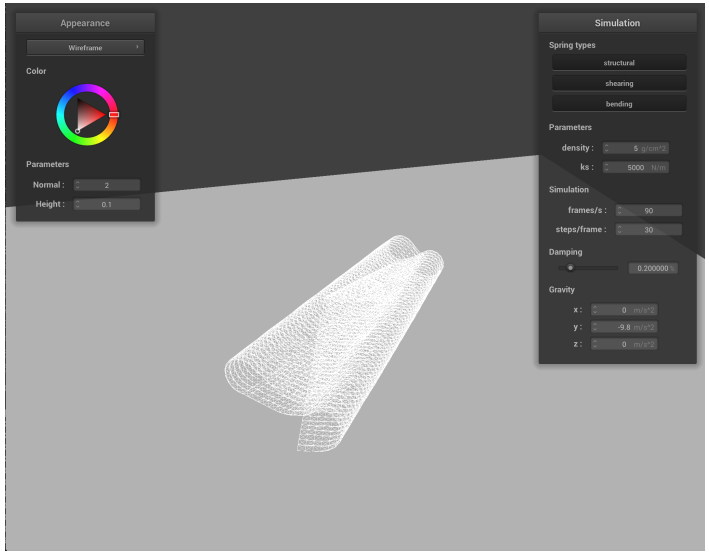

Mid Stage


Late Stage

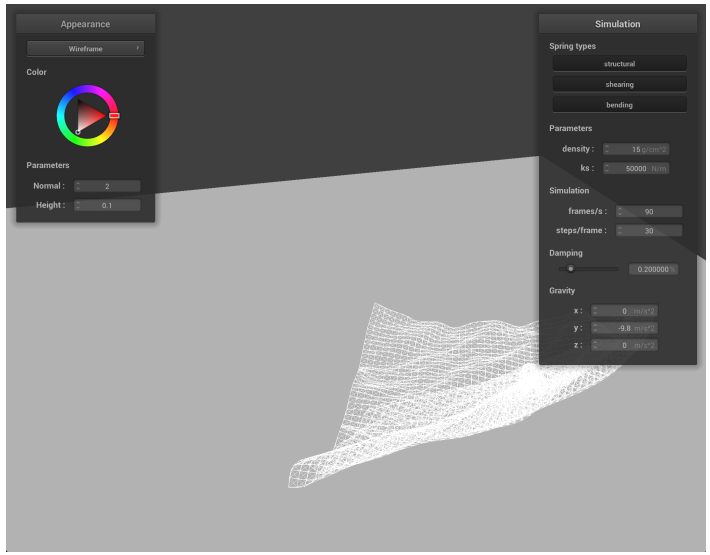
We can also see how changing certain constants, such as density and Ks, affect folding. Since higher densities mean heavier clothes, these clothes fold more sharply; as we see it fold, we can see it has some more weight to how it collapses on itself. A smaller density results in a cloth that folds more lightly, and the folds are larger and overall contain more space and "air."
Clothes with a high Ks have large intra-cloth forces, which causes them to come to rest after a much longer period of time. As the cloth bends and folds, the spring forces work to move the cloth dynamically, so many folds occur that change the landscape of the cloth for a long time. A lower Ks means the forces are small, so the cloth falls tamely due to the external forces (gravity). It tends to fall flat and fold neatly since the cloth lacks strong interactions between its pointmasses.



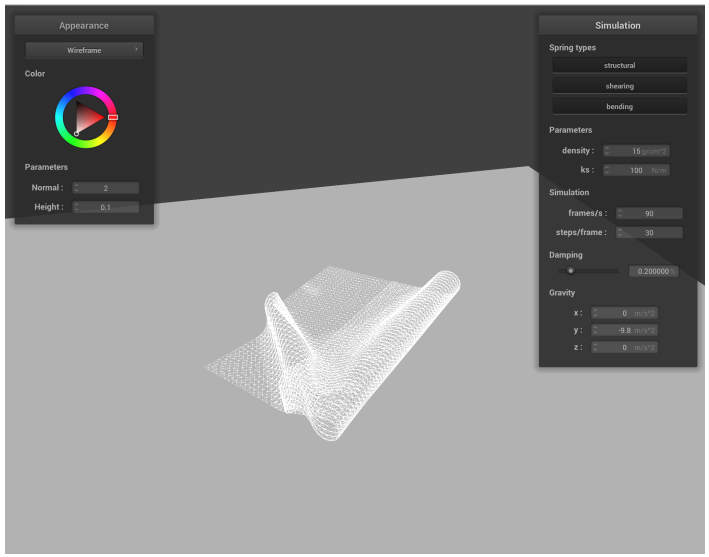
High density cloth.



Low density cloth.



High Ks cloth.



Low Ks cloth.