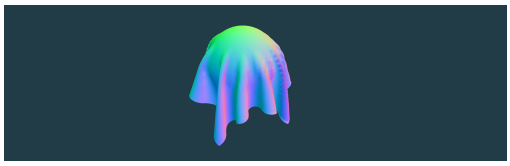
CS184/284A Spring 2025 Homework 4 Write-Up

Names: Andy Zhang

Link to webpage: https://github.com/cal-cs184-student/hw-webpages-zhangnd16
Link to GitHub repository: https://github.com/cal-cs184-student/sp25-hw4-durer

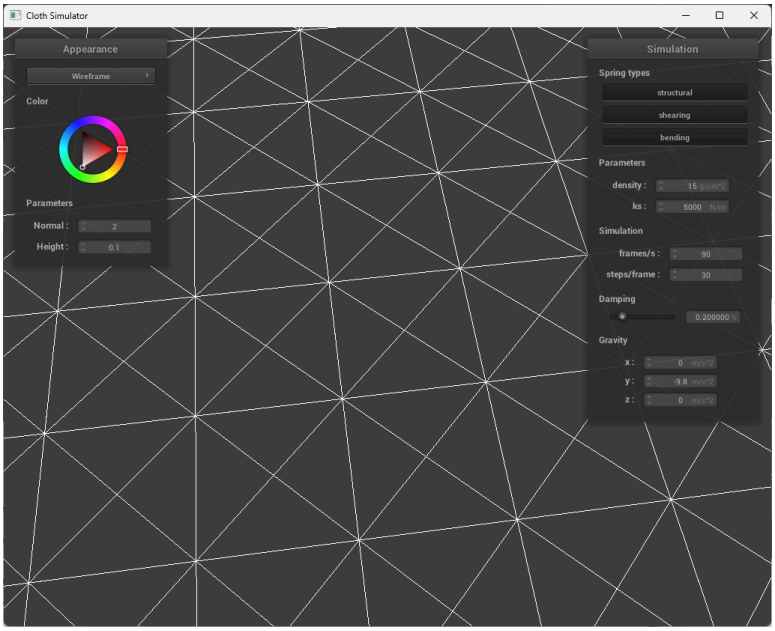


Overview

Homework 4 primarily focuses on the various techniques of physical simulation. With a collection of point masses and springs of different purposes, this project explores ways to simulate this system of point masses as cloth, how it reacts with forces, as well as how it interacts with other primitives and itself. In the last part of this project, some methods of shading models and texture mappings are used to simulate a realistic lighting with the cloth built with the point masses and sphere. Together these techniques serve as a comprehensive realistic simulation system that mimics real world cloth behavior and lighting.

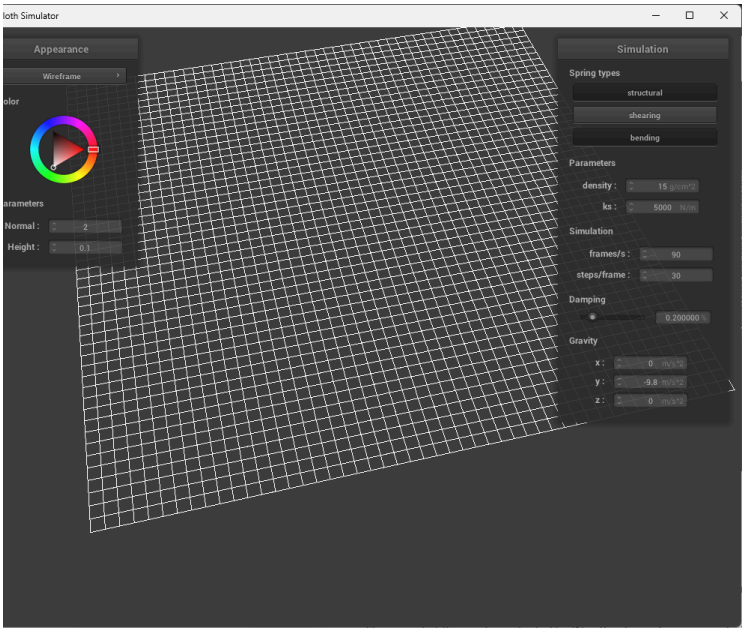
Part 1: Masses and springs

Here is a screenshot of pinned2.json at a close viewing angle.

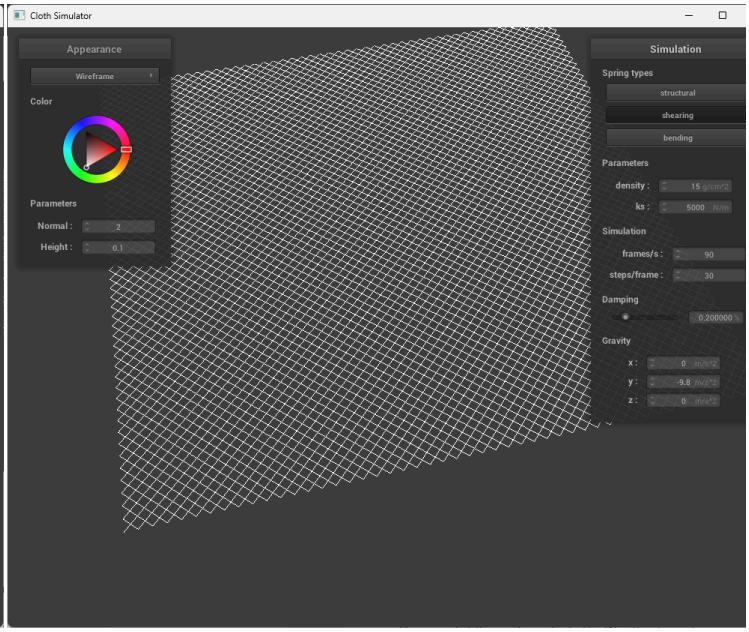


pinned2.json

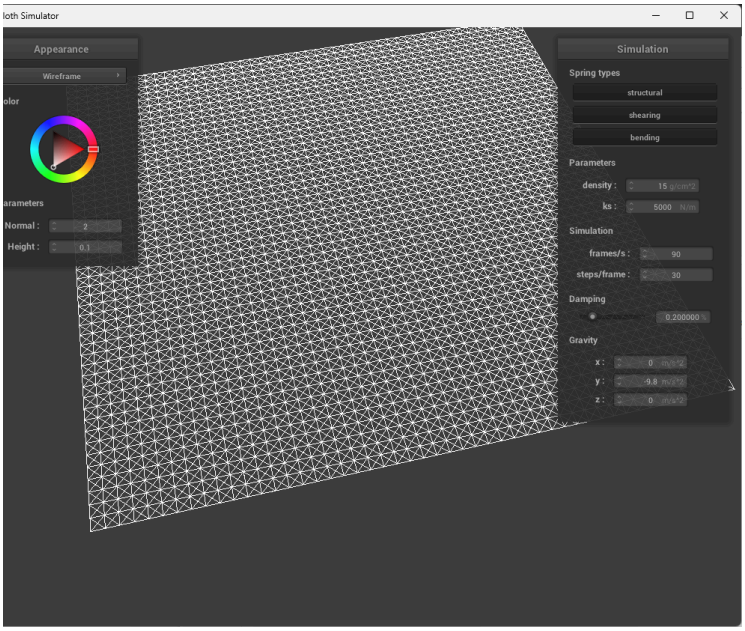
Below are some views of scene/pinned2.json with certain constraints.



No shearing constraints



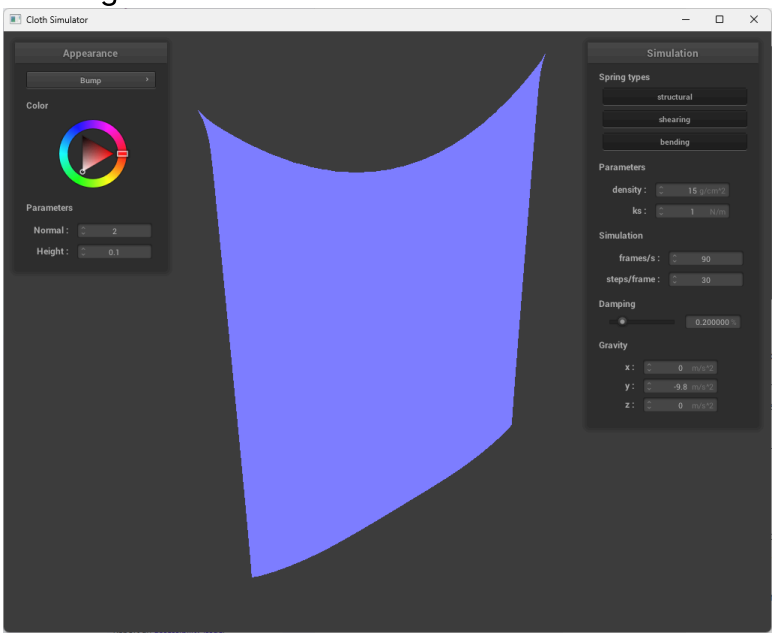
Only shearing constraints



All constraints

Part 2: Simulation via numerical integration

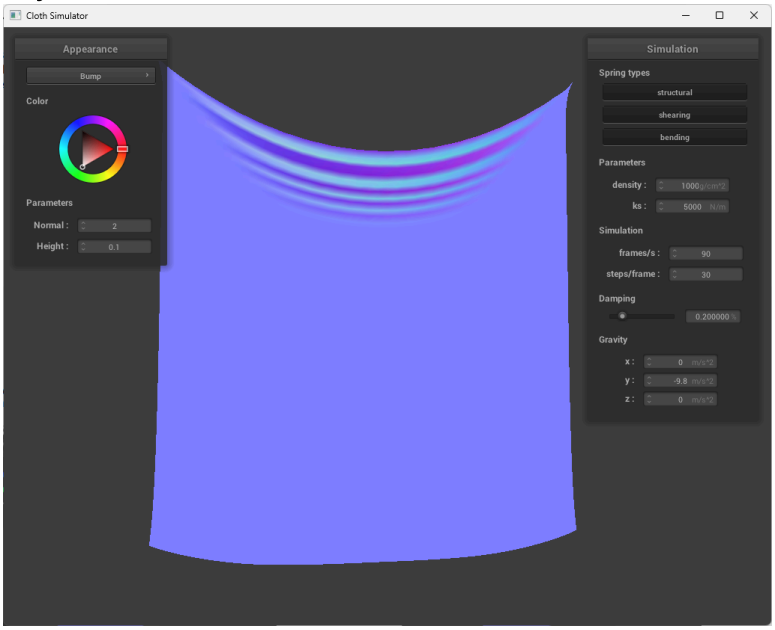
Observing the simulation with differing parameters, a very low ks makes the texture of the cloth very sloppy and loose, with a lot of wrinkles, where as a high ks makes the surface of the cloth more smooth with less wrinkles. A cloth with ks = 1 seems like a film as little internal forces are dragging masses together.



ks = 1

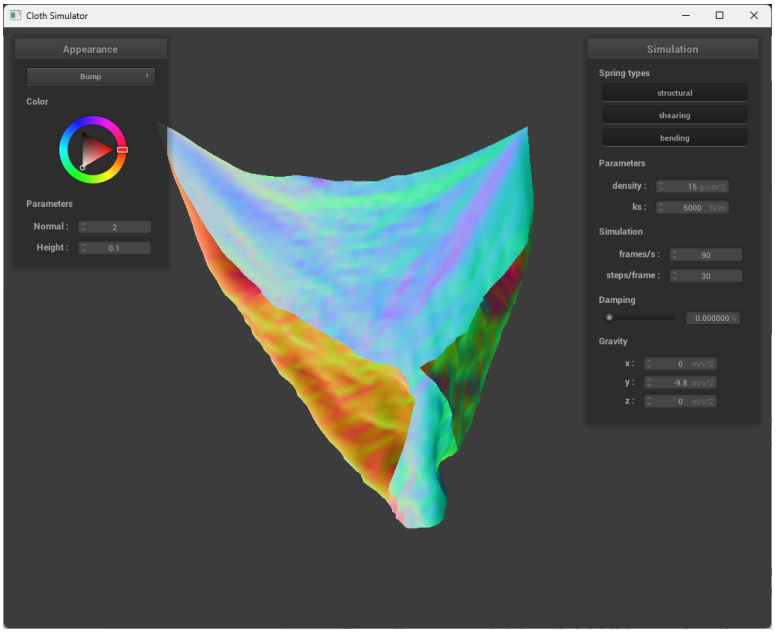
A higher density would make the cloth appear to be more heavy, such that the bottom edge of the curvy becomes curvy by the effect of gravity. A lower density makes the

edge less curvy, as well as less wrinkles on the upper part of the cloth. A cloth with density = 1000 seems very heavy, with a curvy border at the bottom dragged by a large gravity.



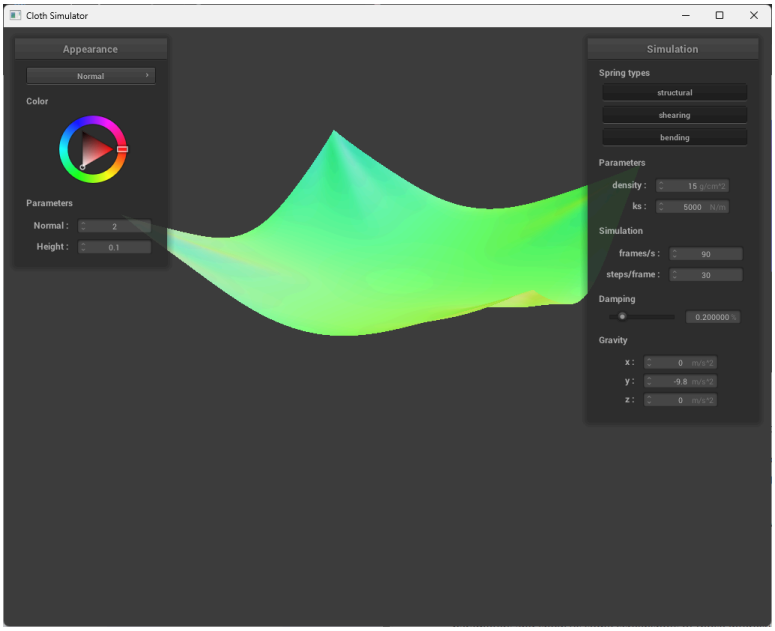
density = 1000

The motion of a low damping cloth is very drastic and takes a long time to fully stop, whereas a high damping cloth is pulled by the gravity very gently. Below is the motion of a 0 damping cloth, as there is no simulated friction/heat loss in the calculation of forces.



density = 1000

Below is the fully stopped pinned4 cloth.



pinned4.json with default parameters

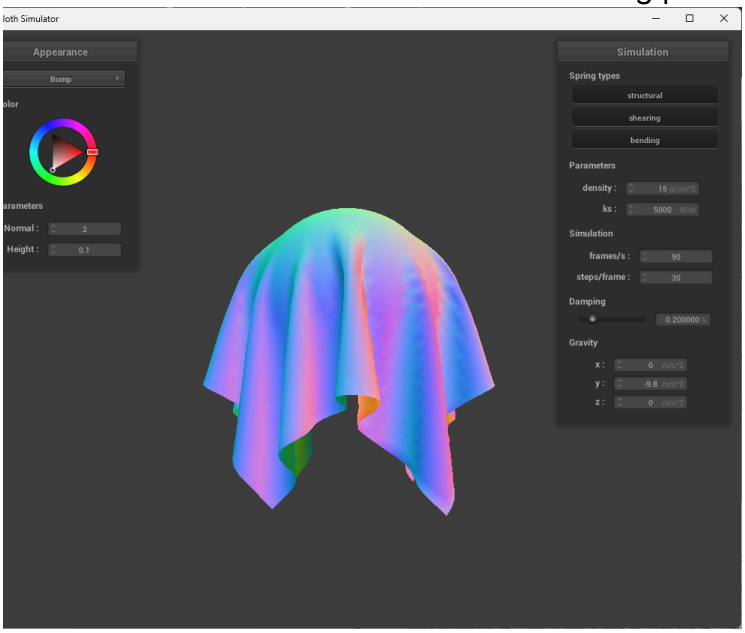
Part 3: Handling collisions with other objects

To handle the collision between a point mass and a sphere, first check whether the point mass is within or intersects with the sphere, that is, whether the distance between the radius of the sphere and the position of the mass is less than or equal to the radius. If yes, calculate the tangent point on the sphere on the ray from the center of the sphere to the position of the point mass, then apply a 1 - friction coefficient to the displacement from last position to the tangent point and add it to the last position to get the new position.

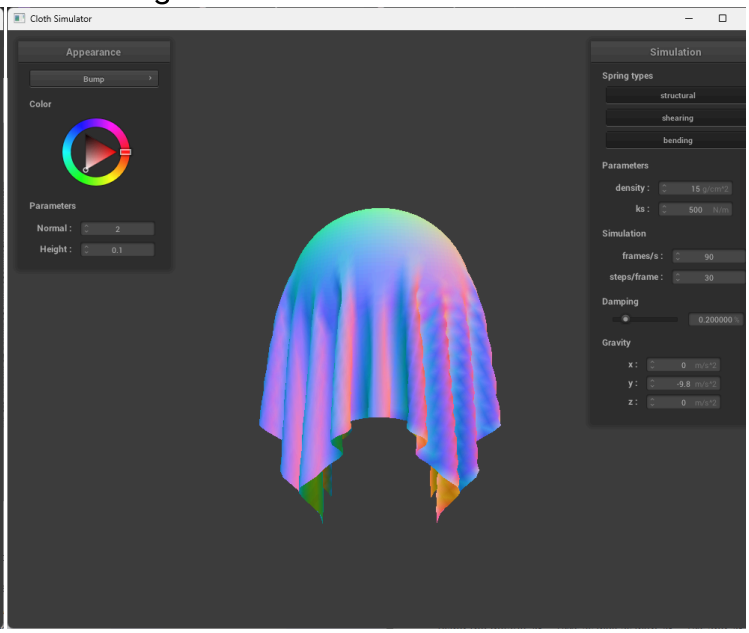
To handle the collision between a point mass and a plane, first check whether the last position and new position are on the different sides of the plane, by taking the dot product of the normal vector of the plane, and the difference between the position vector and a point on the plane. If they have different signs, calculate the intersecting time t using ray-plane intersection formula. Then calculate the displacement from the last position to the intersection point but apply a small constant amount of displacement back in the opposite direction to ensure that the cloth does not overlap with the plane. Add that displacement with friction coefficient to last position to get the new position.

With both collision methods implemented as above, for each time displacement, check whether there is a collision between for every point mass and every primitive.

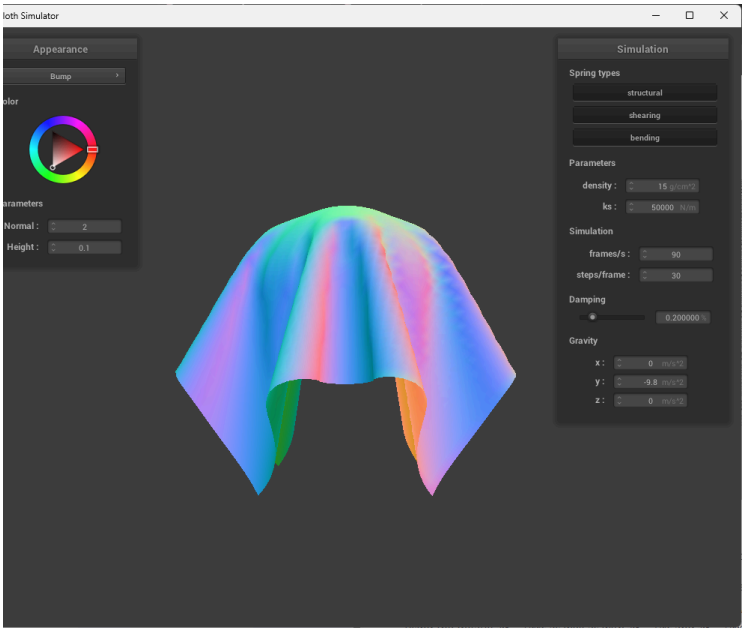
Below are the final resting state of sphere.json and plane.json. Clothes with lower ks feels more loose and sloppy, whereas clothes with higher ks feels more tight as more forces are holding point masses together.



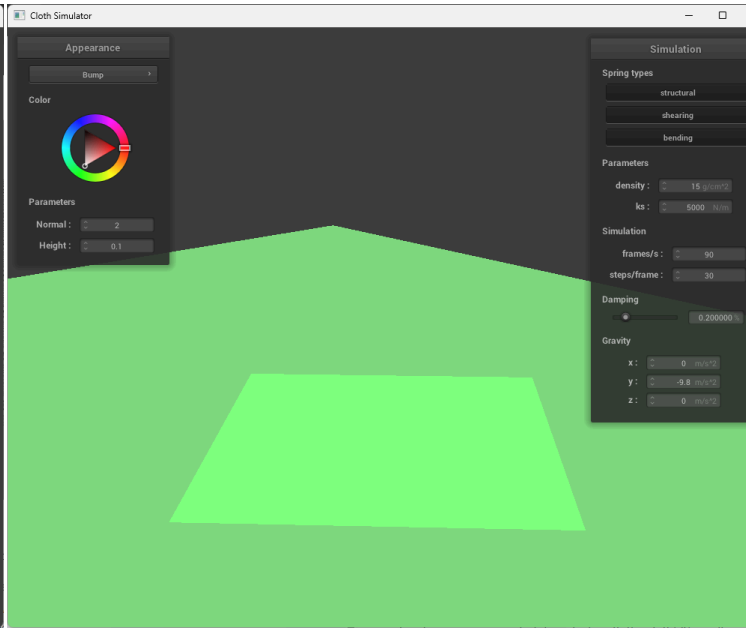
sphere.json with ks = 5000



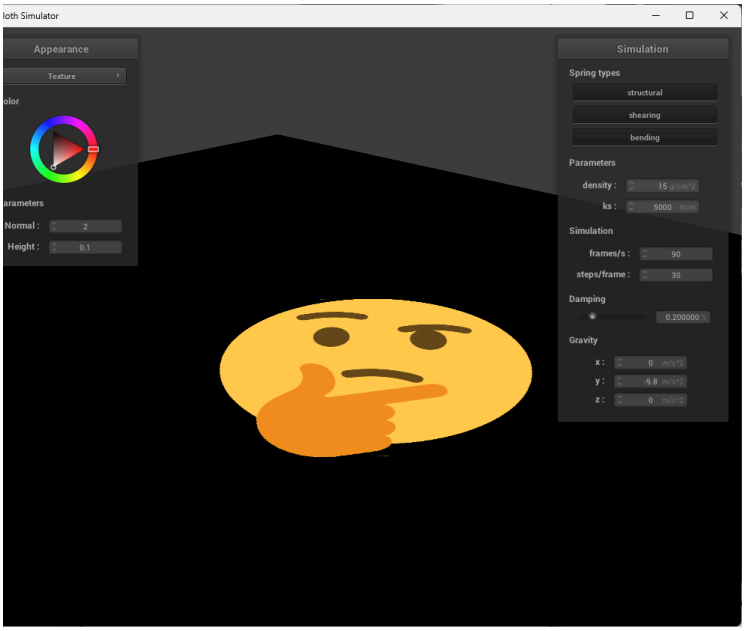
sphere.json with ks = 500



sphere.json with ks = 50000



plane.json



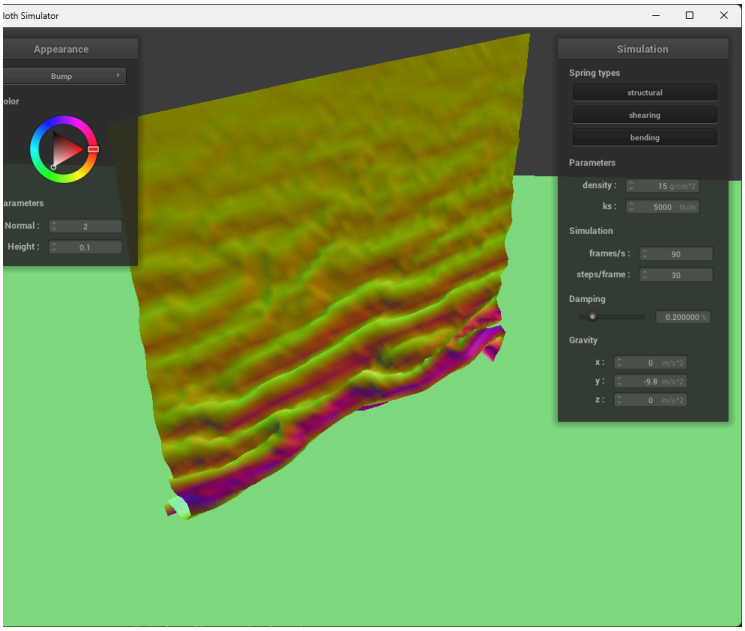
plane.json with thinking emoji texture

Part 4: Handling self-collisions

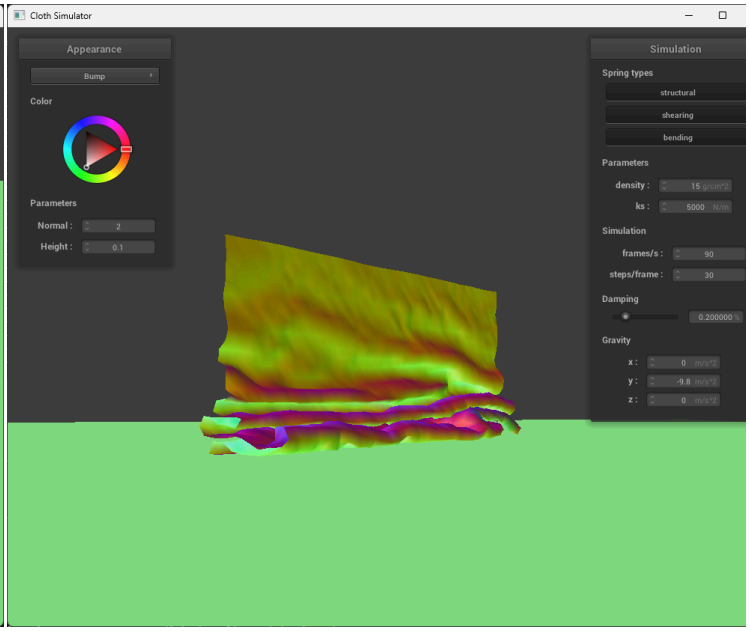
To optimize the time performance of checking the collisions between two point masses, first separate all point masses into a map of vectors with a corresponding hash bounding box, by their hash value using w, h, t in the spec. My hash value is ((pos.x // w) * 10000 + (pos.y // h) * 100 + (pos.z // t)) % 1007. Then, for a given point mass, only check for other point masses in the vector with the hash value of the given point mass, which corresponds to the same bounding box that contains the given point mass.

For every time dt, calculate the hash values of all point masses, put these point masses into the map keyyed by their hash value, and check collision for each point mass using the method above.

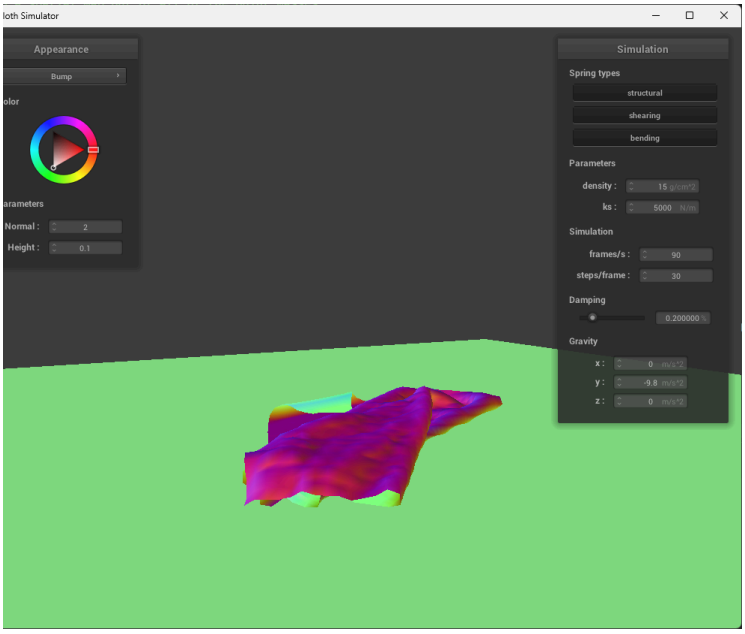
Below are some screenshots of selfCollision.json cloth falling and folding on itself.



Initial state of selfCollision.json

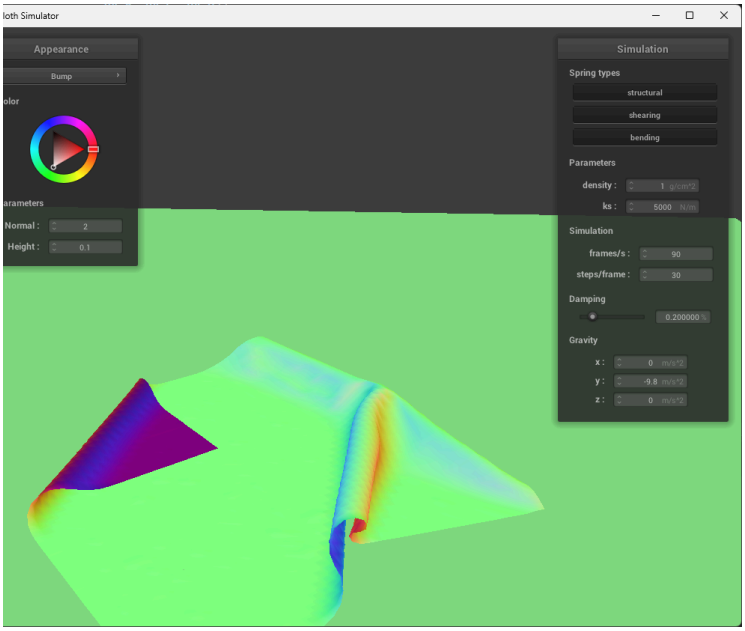


Intermediate state of selfCollision.json

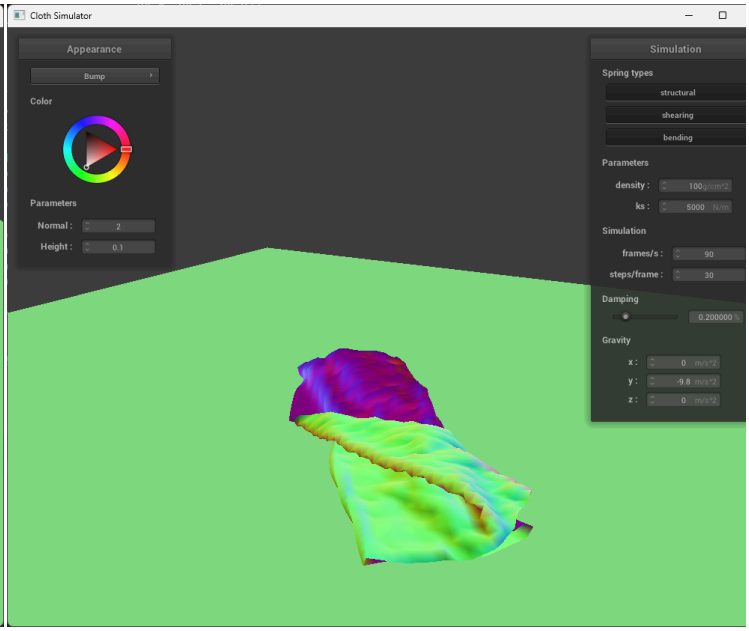


Final resting state of selfCollision.json

Clothes with higher density are dragged harder by the gravity, making them harder to unfold itself. Clothes with lower density tend to make a greater gap between itself, as a less gravitational force has a less interference on the repulsion between point masses.

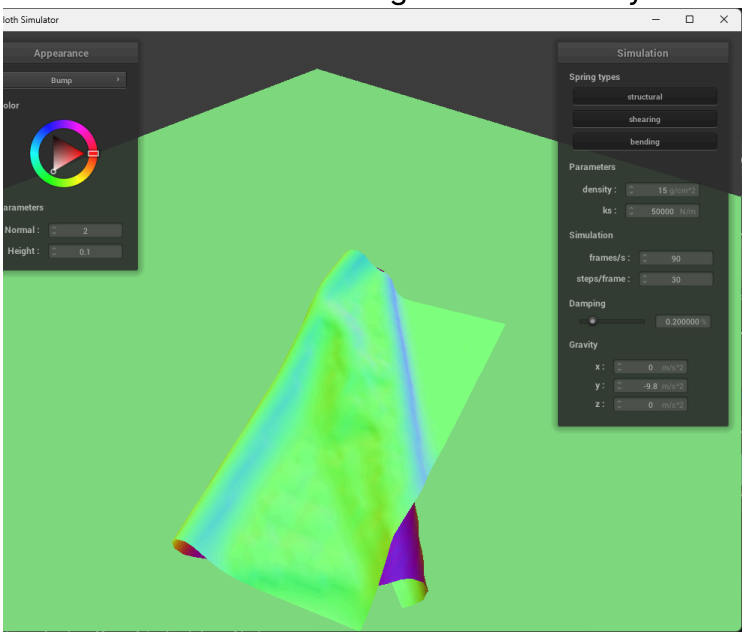


density = 1

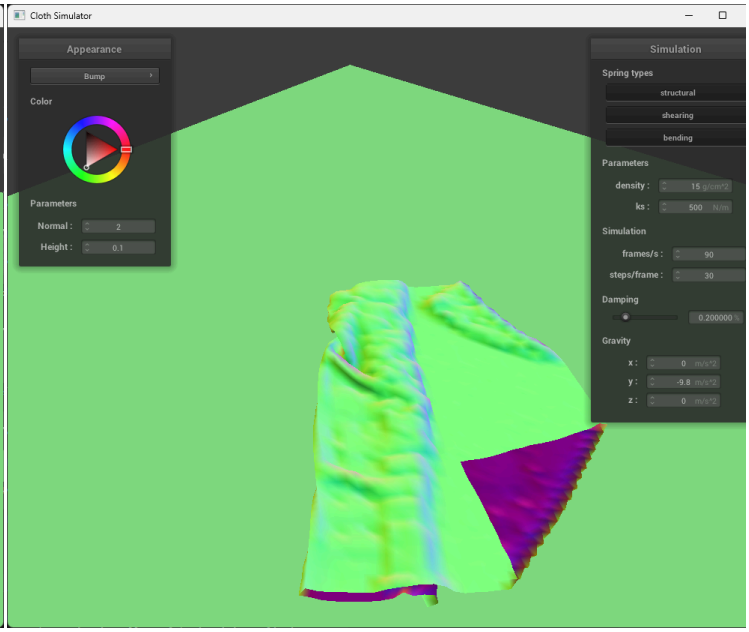


density = 100

Clothes with a higher ks has a stronger connection force between point masses and therefore the resting state has a smoother surface, whereas clothes with a lower ks has a resting state with many wrinkles in it.



ks = 50000



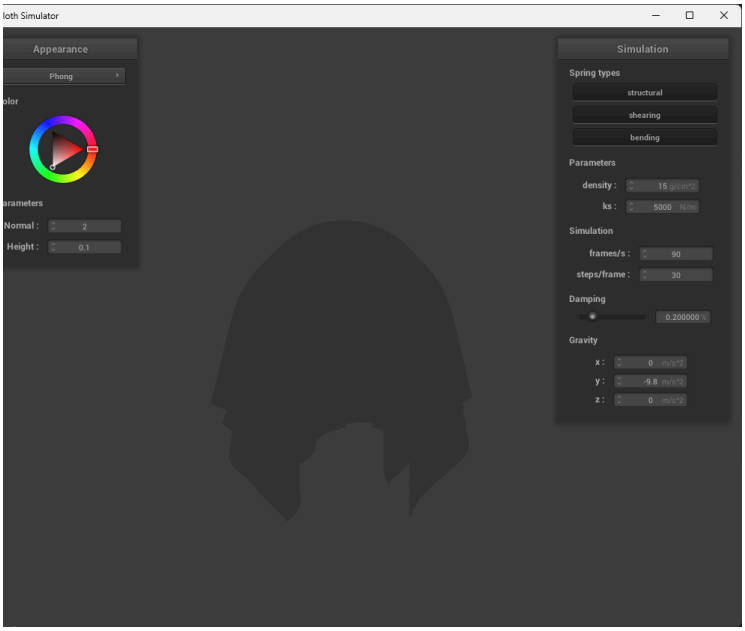
ks = 500

Part 5: Shaders

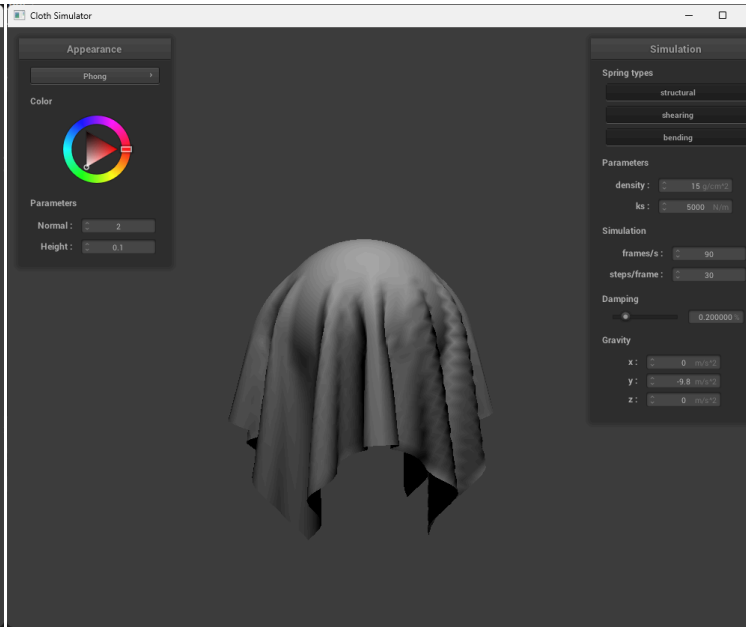
A shader program is a piece of code that is intended for GPU to process parallel computations for graphics to achieve a better time performance. Shaders are especially useful in real time applications, as they are much faster than CPU computations. Vertex shaders are to perform operations on vertices of a model, manipulating the geometry, while fragment shaders are to compute the color on the screen according to the manipulated geometry.

Blinn-Phong shading model is a model to simulate the realistic lighting reflection. Blinn-Phong shading composes of three parts: ambient lighting, diffuse reflection, and specular shading. Ambient lighting casts a uniform lighting to the object regardless of the rays; diffuse reflection uses Lambert's cosine law to calculate the lighting at a viewing direction, which serves as a contrast of lighting on the object at different angles; specular shading simulates a spotlight on the object that is not only dependent on the viewing angle, but also on the light direction. Lastly, Blinn-Phong shading model sums all these three lighting together to get a realistic shading.

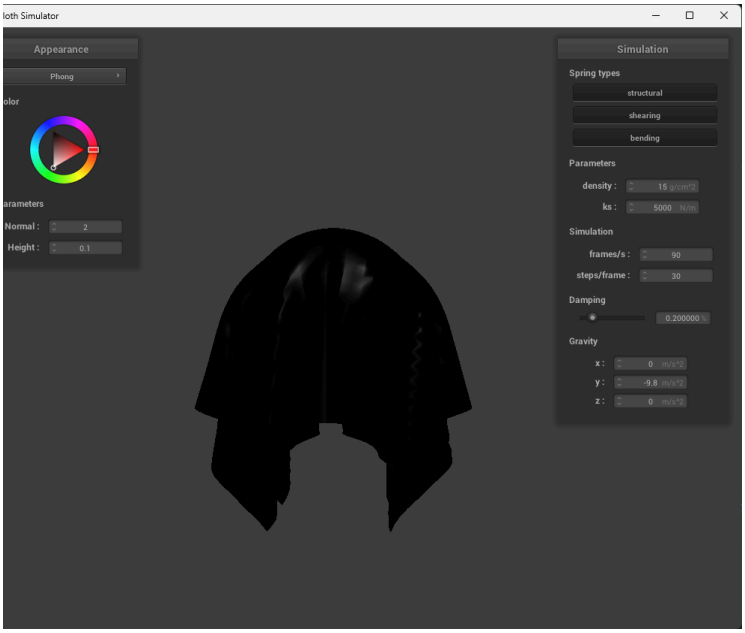
Below are some screenshots that show the ambient, diffuse, specular component of the model, and the entire model that contains all three.



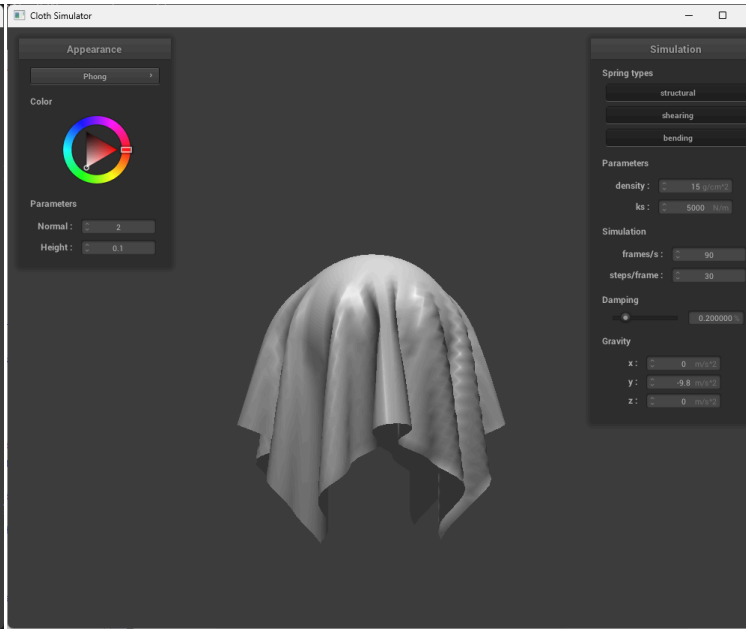
Ambient component



Diffuse component



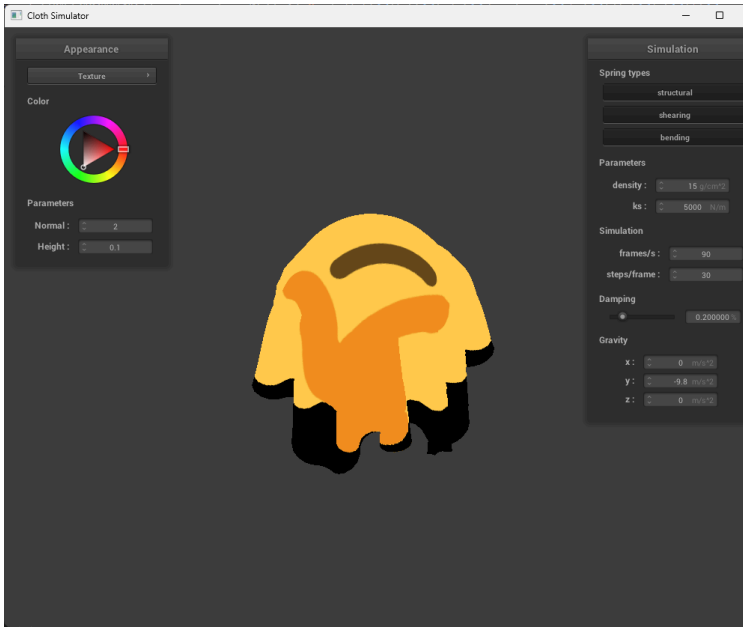
Specular component



Full Blinn-Phong shading

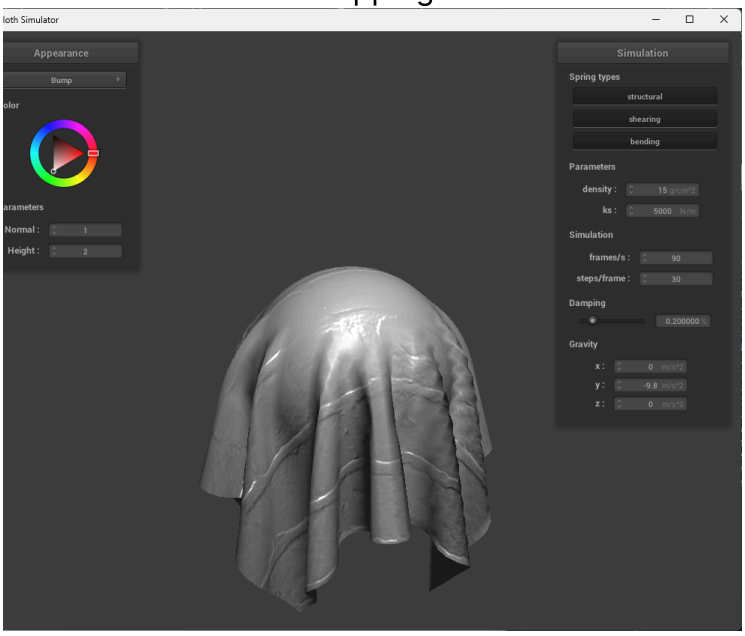
Below is a screenshot of a texture mapping shader with the thinking emoji.

Thinking emoji

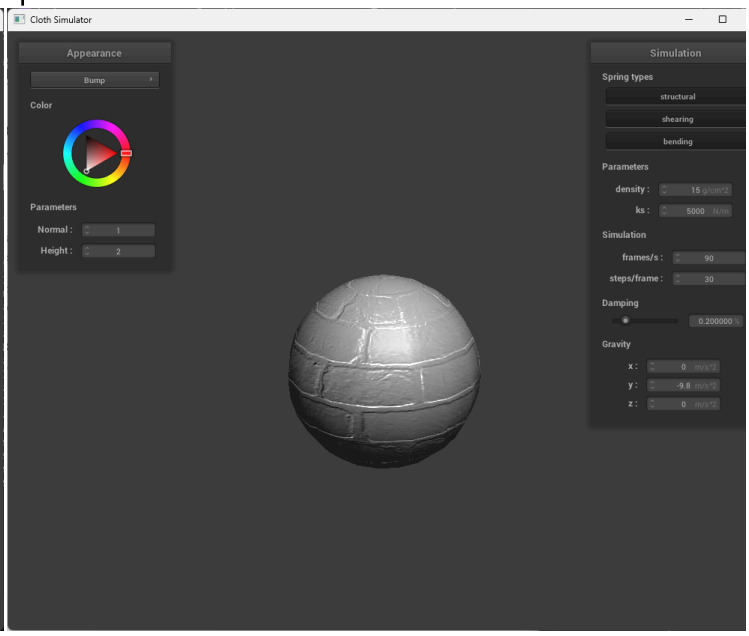


Texture mapping with thinking emoji

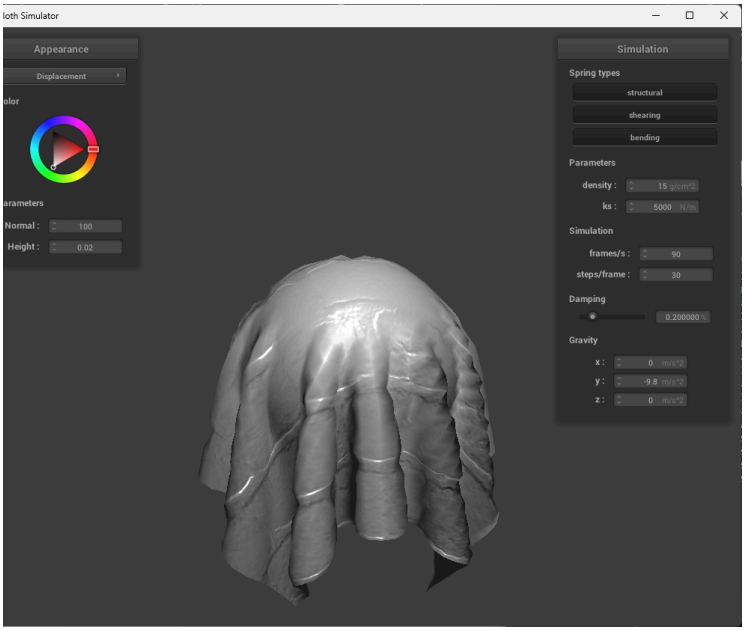
Below are screenshots of bump mapping and displacement mapping on the cloth and the sphere.



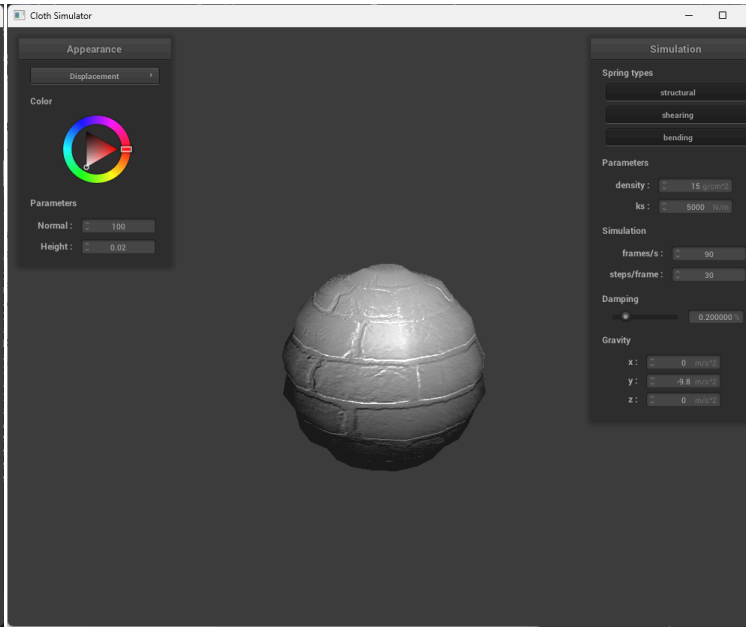
Bump mapping cloth, with -o 16 -a 16



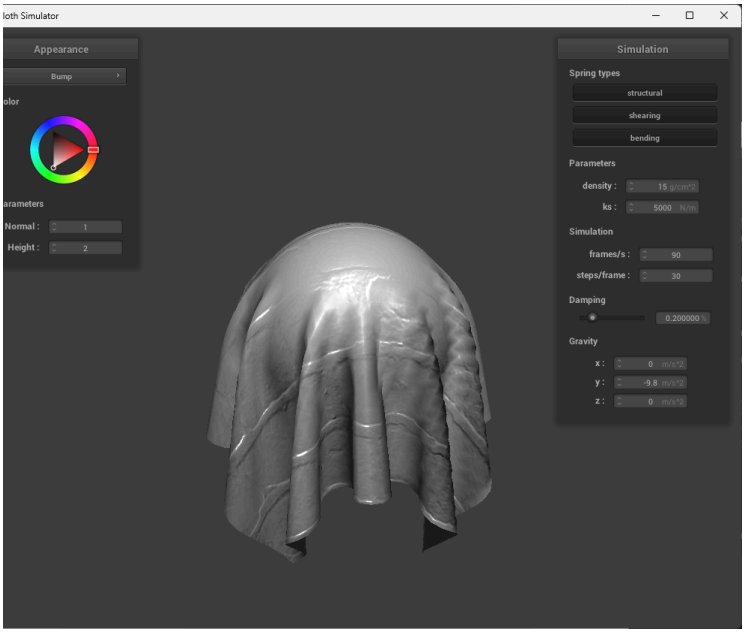
Bump mapping sphere, with -o 16 -a 16



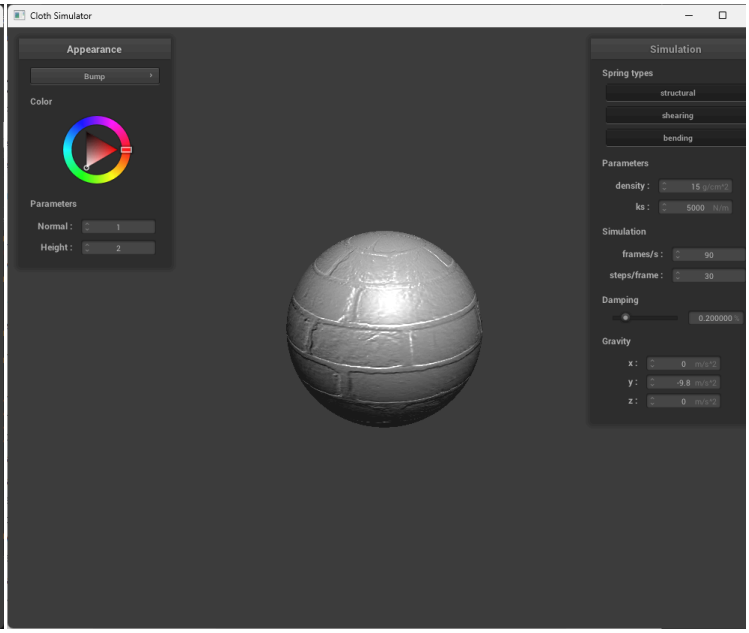
Displacement mapping cloth, with -o 16 -a 16



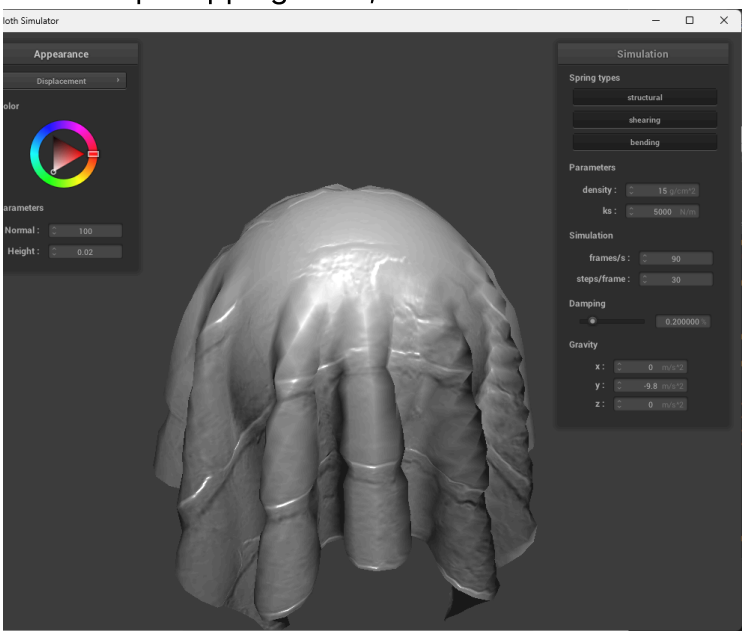
Displacement mapping sphere, with -o 16 -a 16

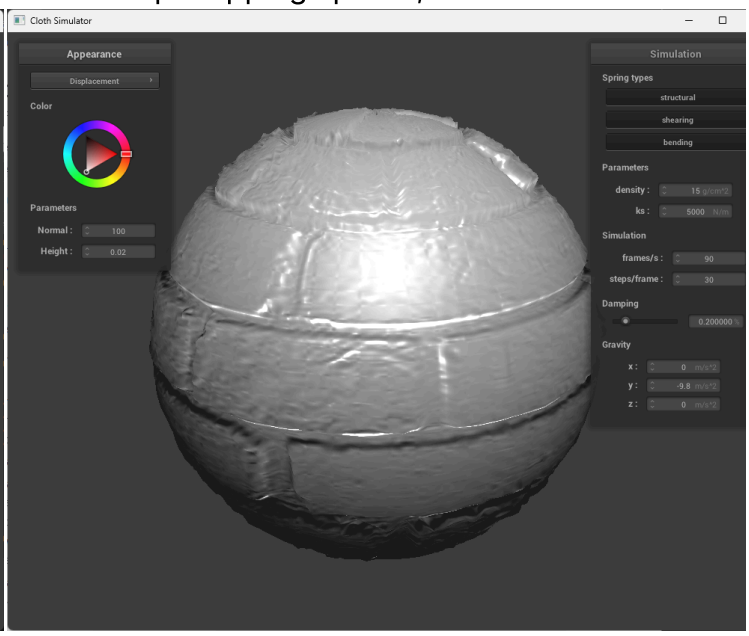


Bump mapping cloth, with -o 128 -a 128



Bump mapping sphere, with -o 128 -a 128



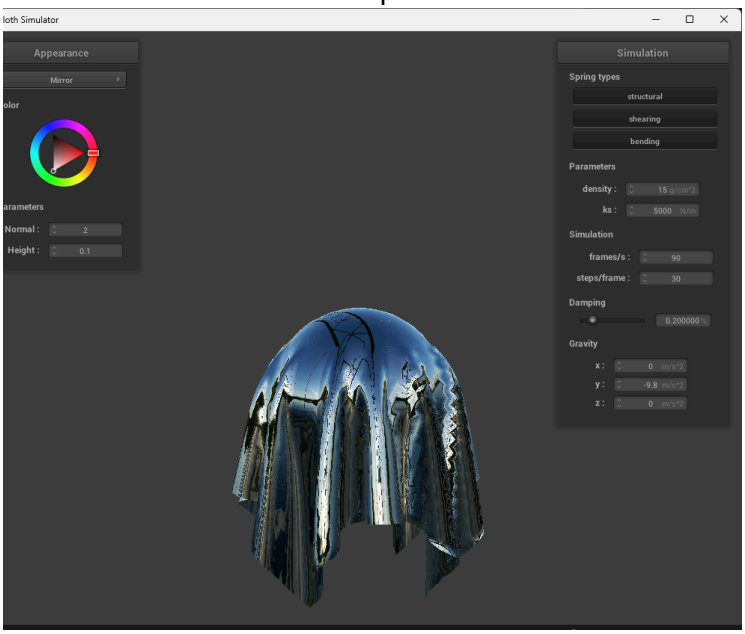


Displacement mapping cloth, with -o 128 -a 128

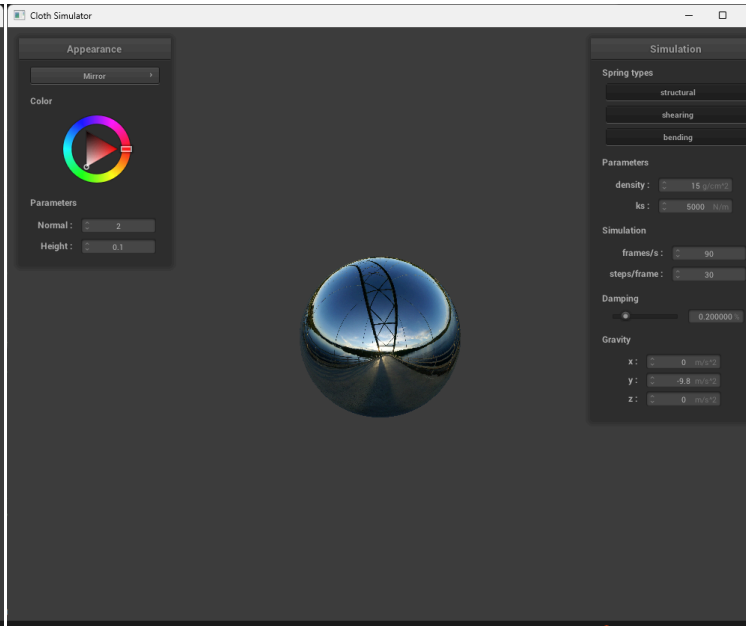
Displacement mapping sphere, with -o 128 -a 128

Bump mapping changes the texture of the surface of the objects. By comparing the screenshots generated by bump mapping and displacement mapping, one can tell that in addition to the bump mapping, displacement mapping adds some manipulation on the vertices of the models that adds some valleys on the surface of the objects, which contributes to the coarseness of the object. This effect becomes less observable as the resolution increases.

Below are screenshots of the mirror shader on the cloth and the sphere.



Mirror shader on the cloth



Mirror shader on the sphere