CS 184: Computer Graphics and Imaging, Summer 2020

Project 4: Cloth Simulator

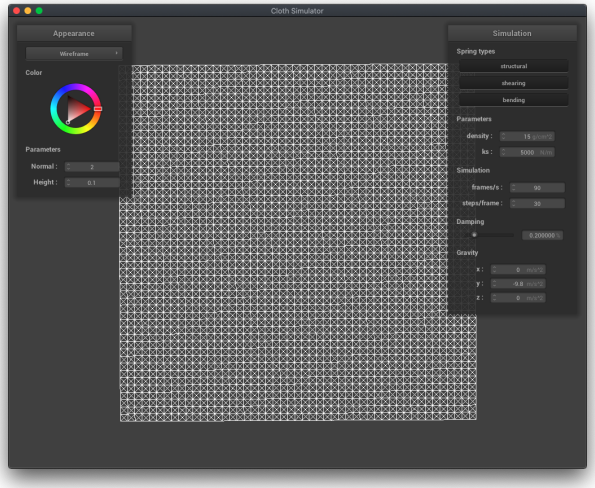
Shikai Qiu

Overview

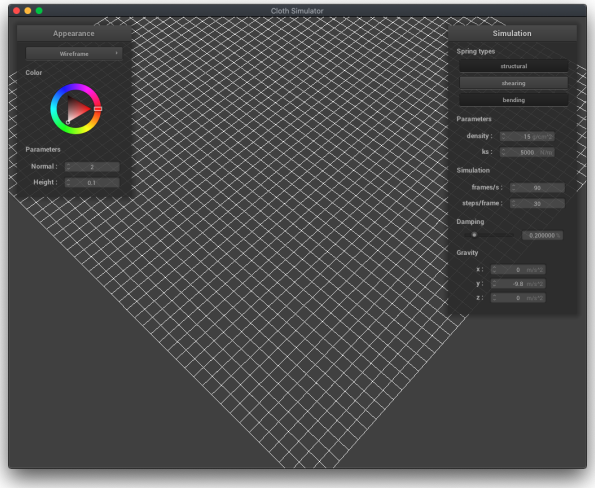
In this project, I implement a real-time simulation of cloth using a mass and spring based system. I build the data structures to discretely represent the cloth, define and apply physical constraints on them, and apply numerical integration to simulate the way cloth moves over time. Furthermore, I implement collisions with other objects as well as self-collisions to prevent cloth clipping. Finally, I implement various shaders that enable more photo realistic rendering of the cloth.

Part I: Masses and springs

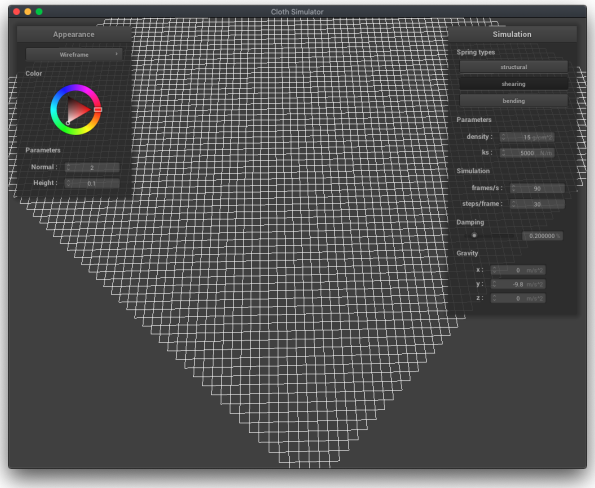
In this part, I implement a function that takes in the cloth's parameters and constructs a representation for its mass distribution and structural constraint with point masses and interconnected springs.



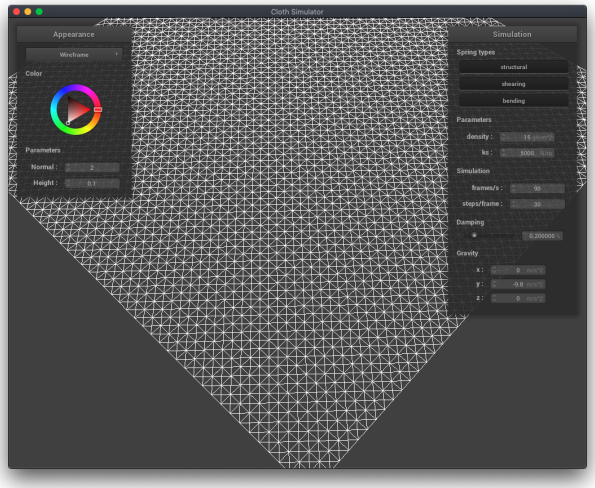
The complete wireframe



The wireframe with only structural and bending constraint



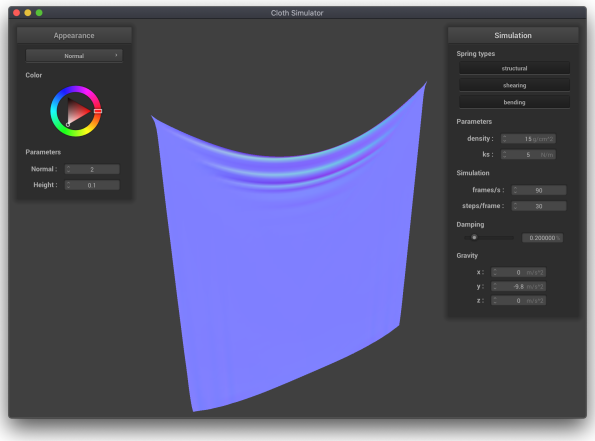
The wireframe with only sheering constraint



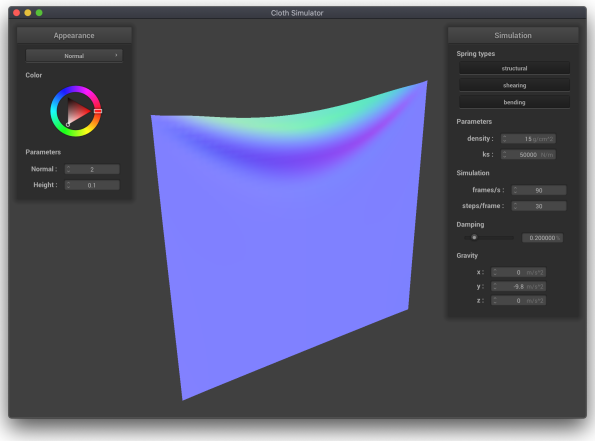
The wireframe with all three constraints

Part II: Simulation via numerical integration

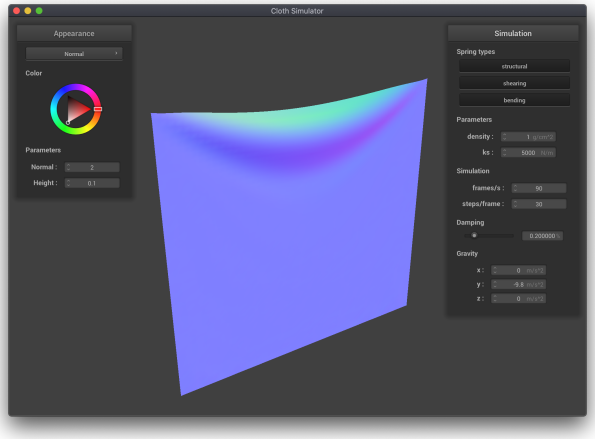
In this part, I implement Position-based Verlet Integration that specifies the motion of the cloth by solving Newton's equation of motion give stability constraints. Below illustrates how parameters such as spring constants, mass density, and damping factor affect the behavior of t cloth.



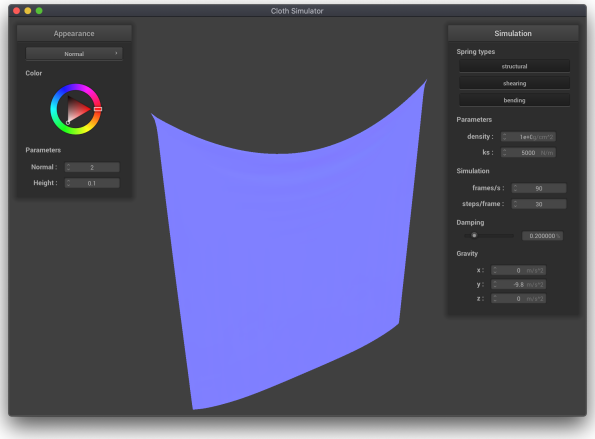
ks = 5 N/m, in equilibrium



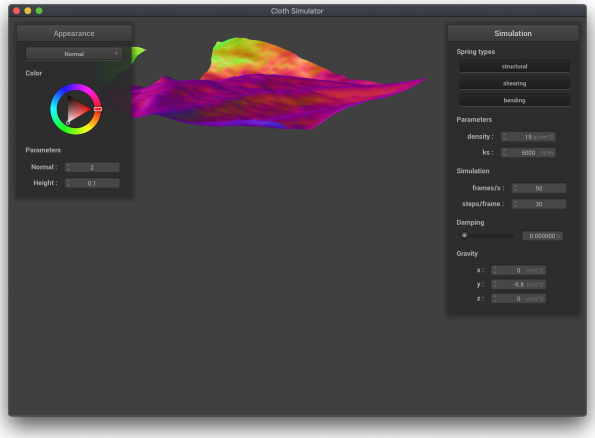
ks = 50000 N/m, in equilibrium



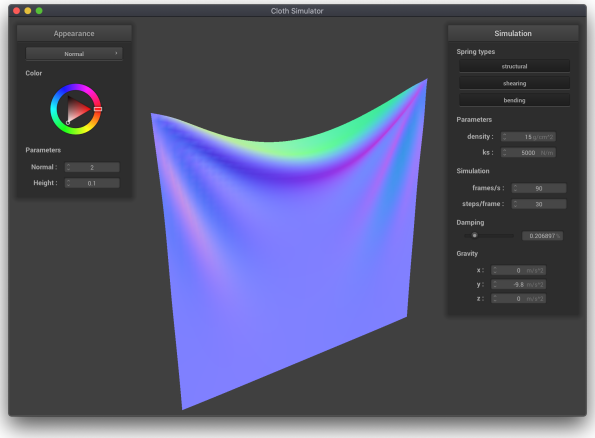
density = 1 g/cm^2, in equilibrium



density = 10000 g/cm^2, in equilibrium



damping = 0%, end of the first oscillation

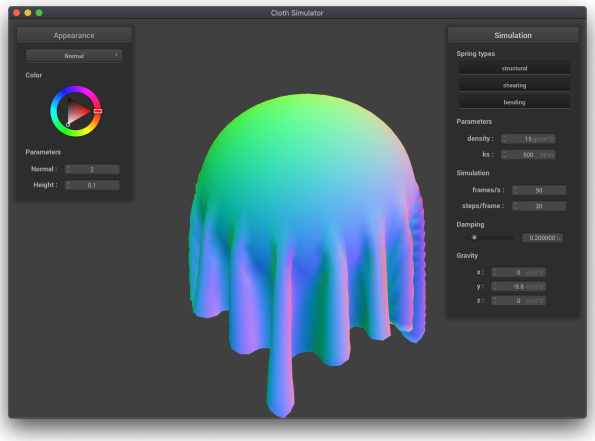


damping = 20%, end of the first oscillation

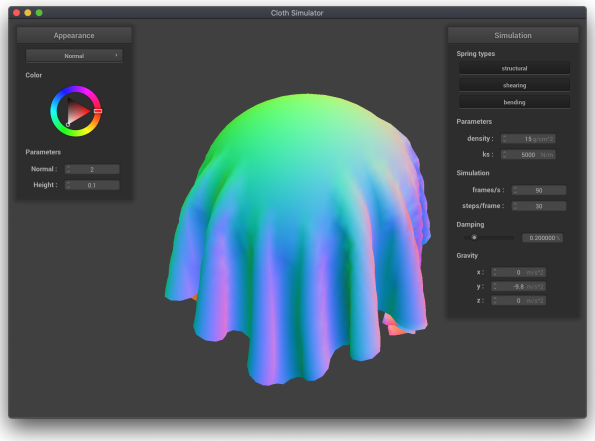
As we can see, increasing the spring constants makes the cloth more rigid in equilibrium, increasing the density makes the cloth elongate more in the downward direction in equilibrium, and increasing the damping factor brings the cloth to stop in fewer number of oscillations.

Part III: Handling collisions with other objects

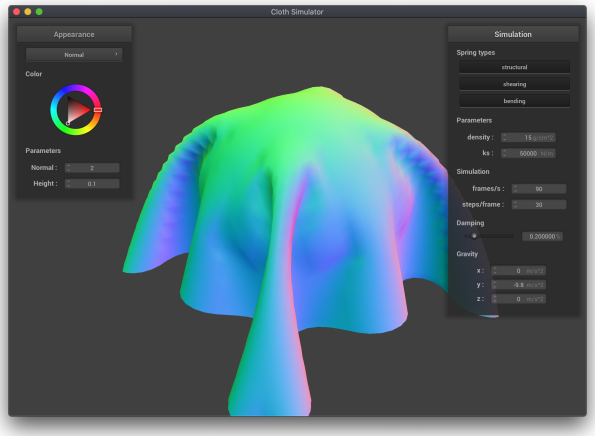
In this part, I add support for cloth collision with other objects in the scene including spheres and planes. The following screenshots illustrate the effects.



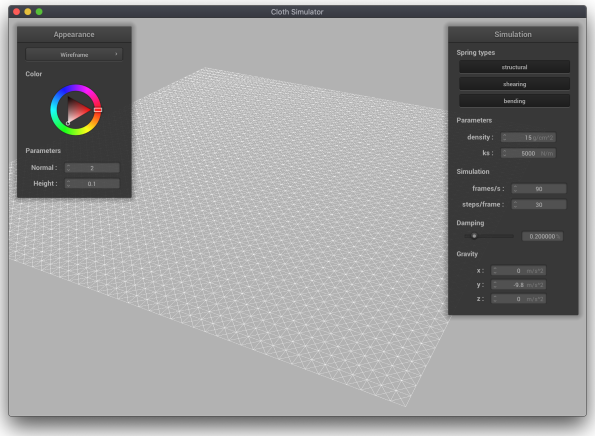
ks = 500 N/m



ks = 5000 N/m



ks = 50000 N/m

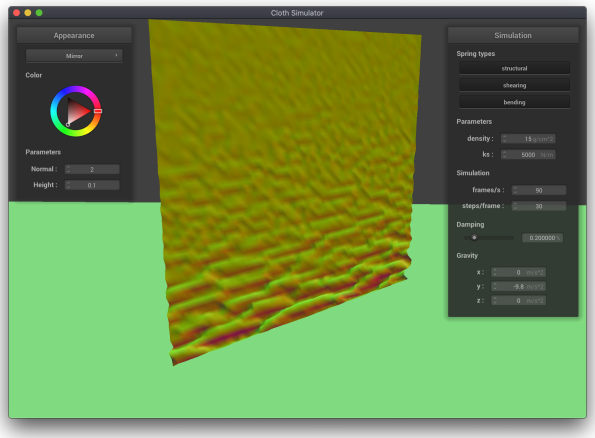


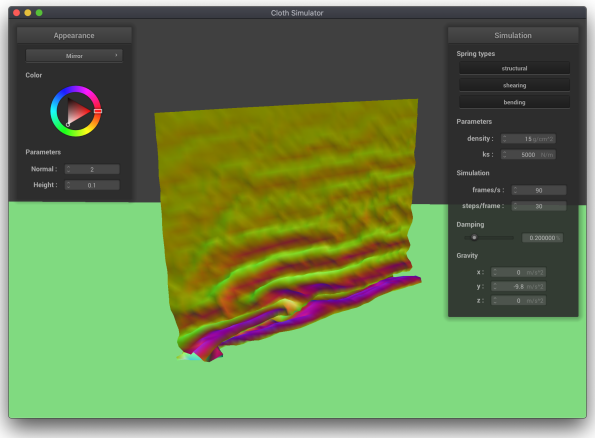
Collision with a plane

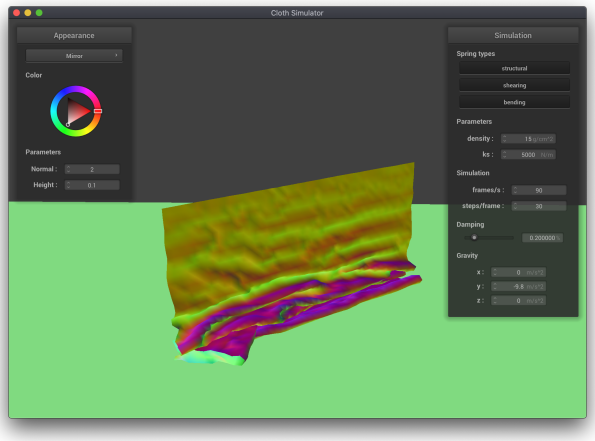
As we increase the spring constants, the cloth becomes more rigid. More precisely, the minimal energy configuration is a compromise between minimizing the gravitational potential energy and minimizing the potential energy of the spring system. Therefore, with higher spri constants, the cloth prefers a less stretched state over a lower center-of-mass state.

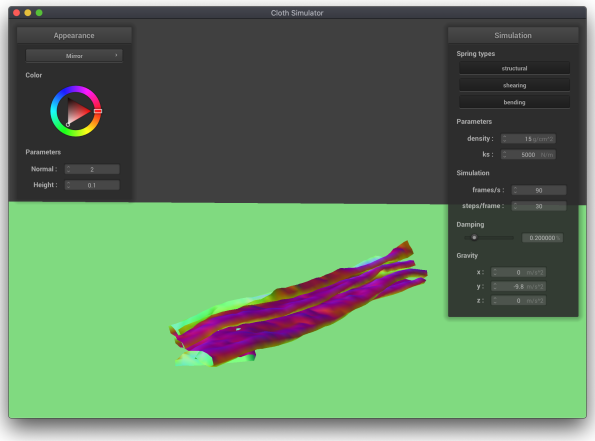
Part IV: Handling self-collisions

In this part, I add support for self-collision of the cloth based on spatial hashing. The effects are shown below.



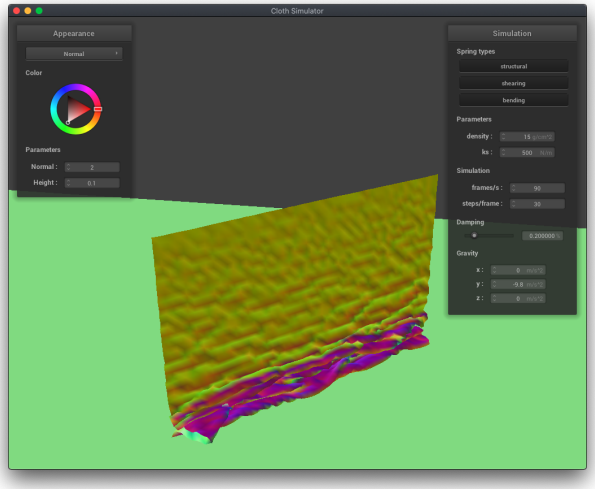


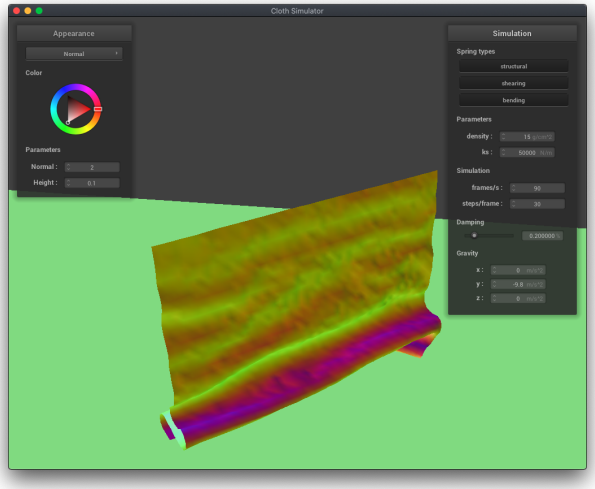




The cloth folds onto itself

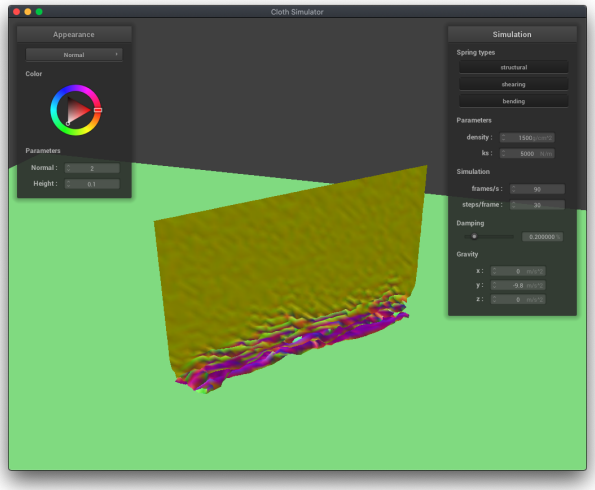
As illustrated below, increasing the spring constant or decreasing the density makes the spring bend less as it falls on to itself because th internal forces of the cloth will overcome the tendency of the cloth to collapse onto itself under gravity.

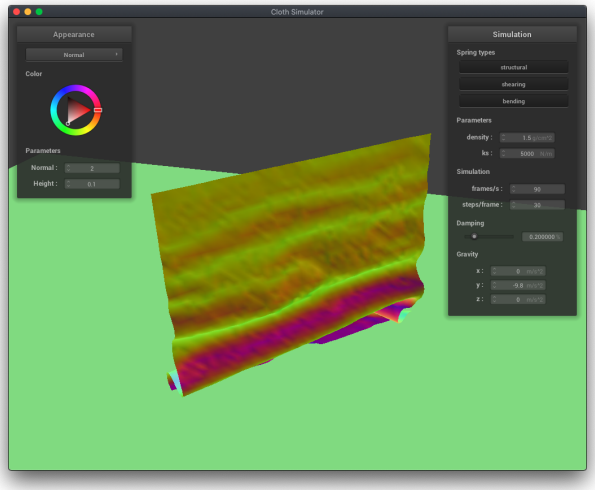




ks = 500 N/m

ks = 50000 N/m





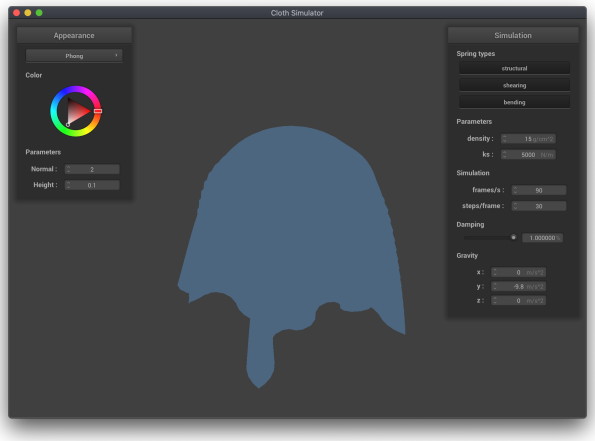
density = 1500 g/cm^2

density = 1.5 g/cm^2

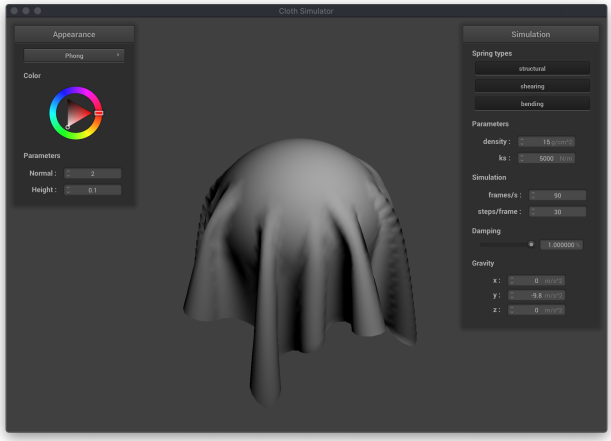
Part V: Shaders

Here I implement several shader programs. A shader program is composed of a vertex shader and a fragment shader. A vertex shader process each vertex and outputs information for the fragment shader to interpolate over each fragments, such as normal vectors and texture coordinates.

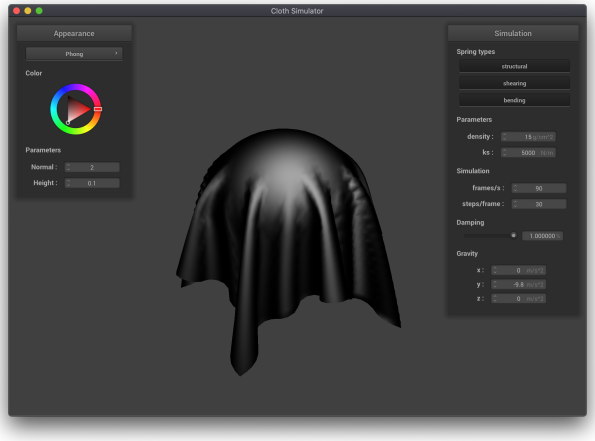
Blinn-Phong shading: This model uses surface normal vector, light position, and viewing direction to calculate diffuse and specular components of the outgoing radiance. The diffuse component is independent of viewing direction while the specular component can be. A additional ambient component is also allowed as an additive constant to the outgoing radiance.



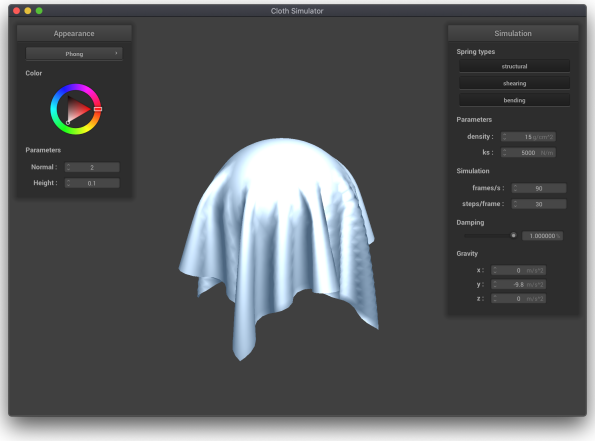
Ambient only



Diffuse only

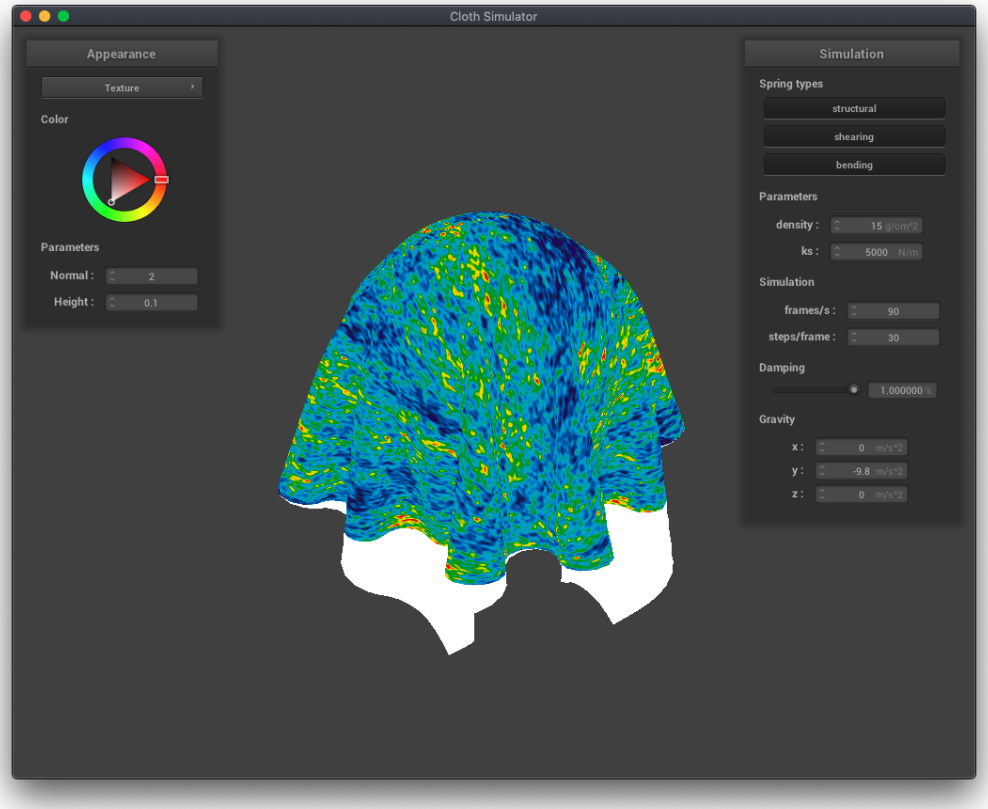


Specular only

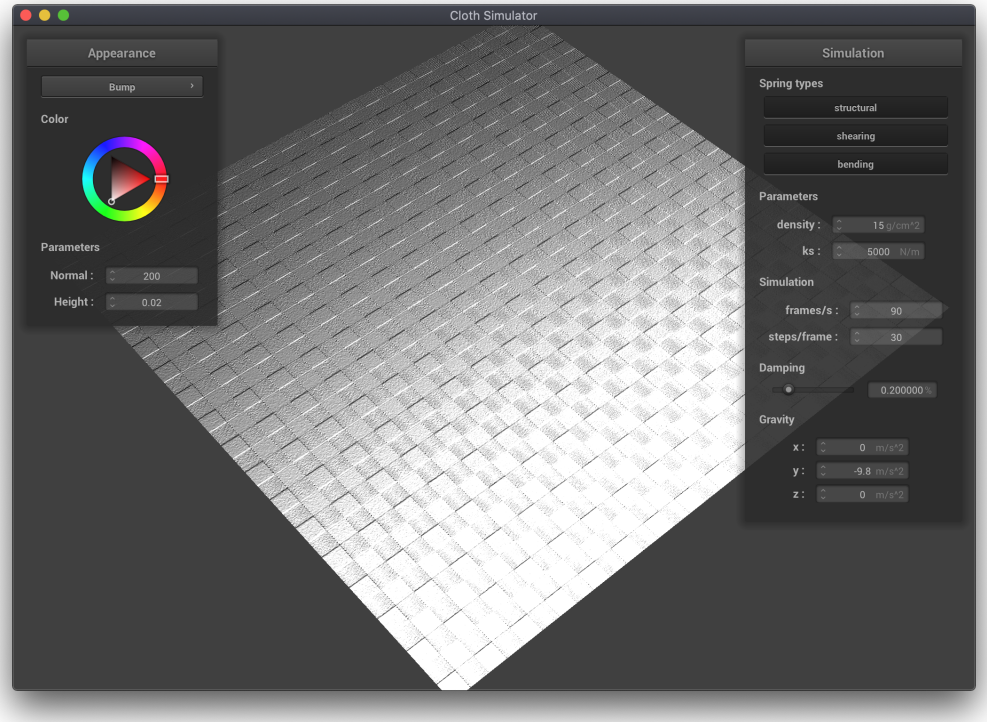


Combined

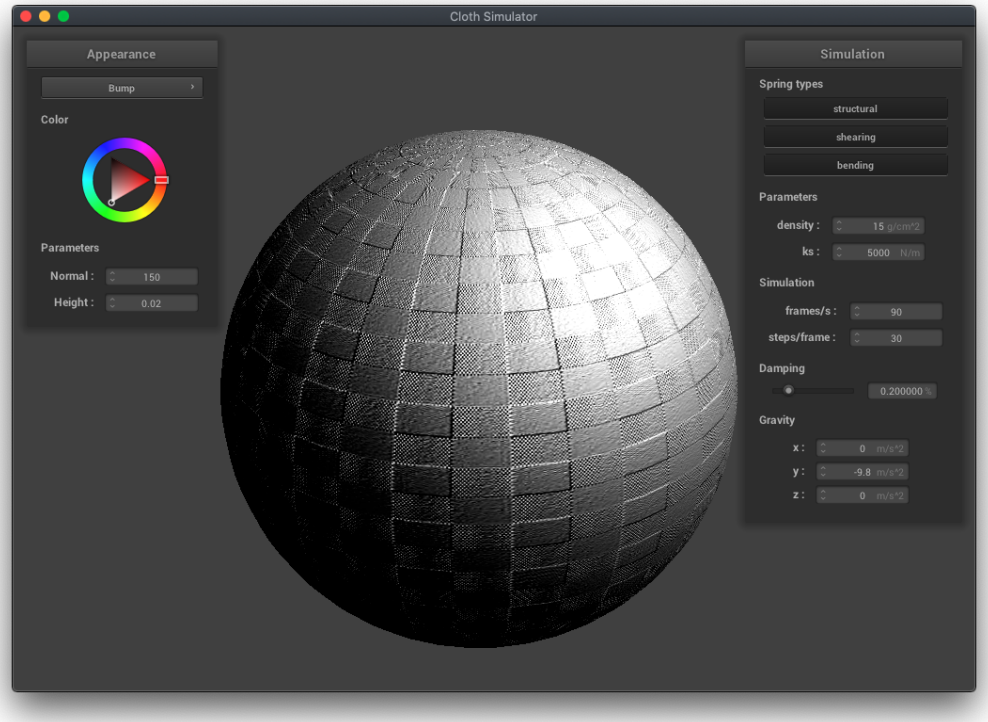
Texture mapping, bump mapping, and displacement mapping are also implemented and shown below.



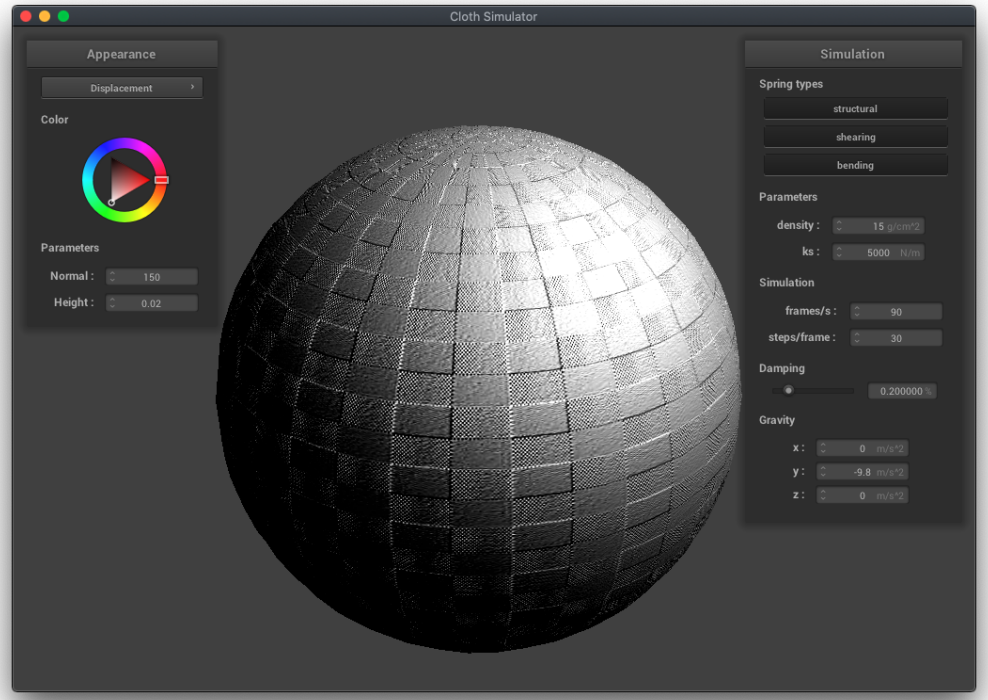
Texture mapping using the Microwave Background Radiation



Bump mapping on the cloth

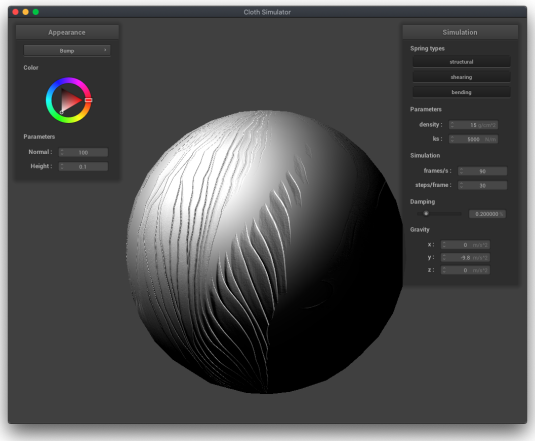


Bump mapping on the sphere

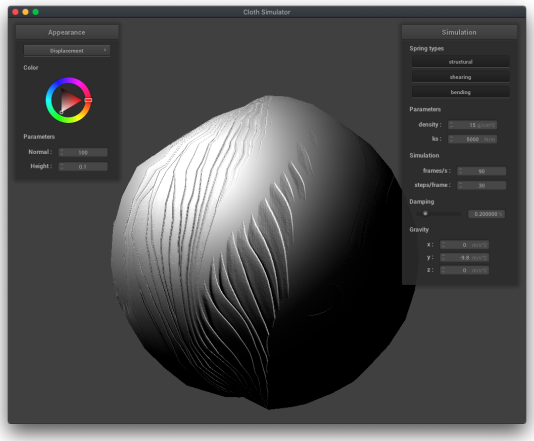


Displacement mapping on the sphere

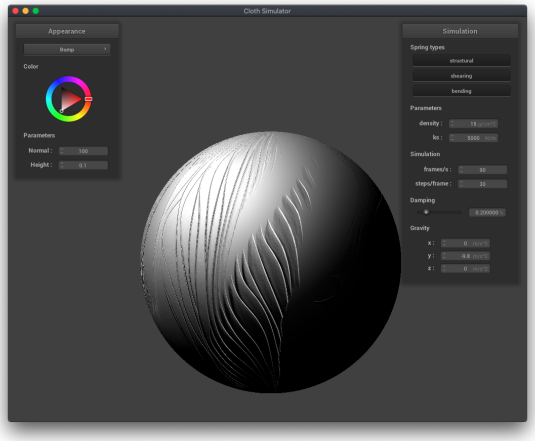
Bump mapping and displacement mapping results in almost identical shading because the normal vector field is the same at each vertex But a closer examination below shows how the two have visibly different effects on the geometry of the object. In particular, when the mes is coarse, the displacement mapping introduces low-frequency deformation to the object that does not accurately reflect the prescribed height field. When the mesh is less coarse, displacement mapping introduces high-frequency deformation to the object while bump mapping leaves the geometry of the object invariant.



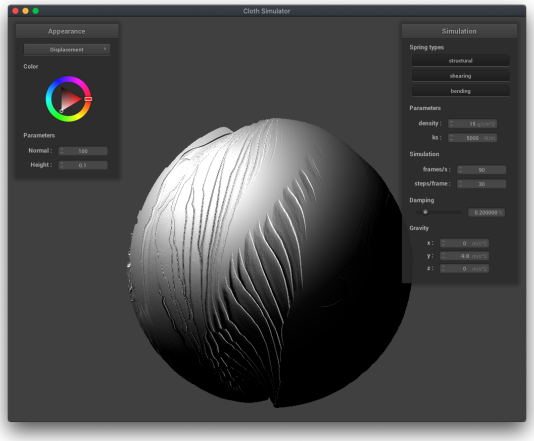
Bump mapping on a coarse mesh



Displacement mapping on a coarse mesh



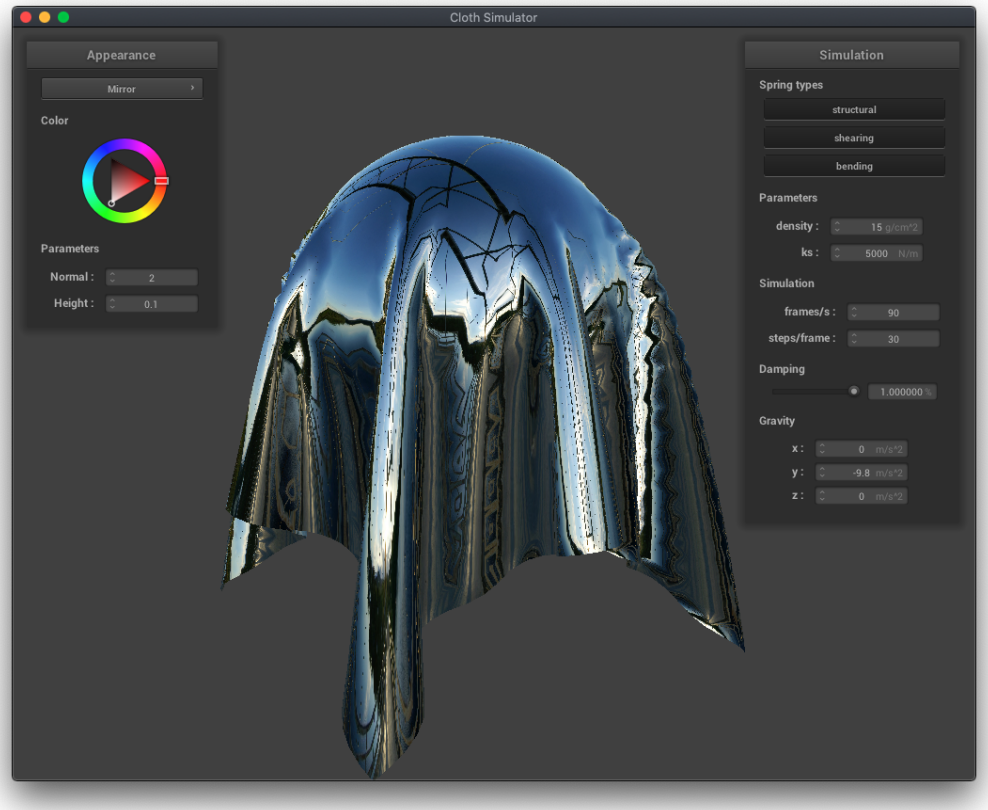
Bump mapping on a refined mesh



Displacement mapping on a refined mesh

Finally, I implement the mirror shader whose effect is shown below.

Mirror shading on a sphere



Mirror shading on a cloth