url: https://github.com/D7ry/cs184-proj-writeup/blob/master/proj4/writeup.md

CS 184: Computer Graphics and Imaging, Spring 2023

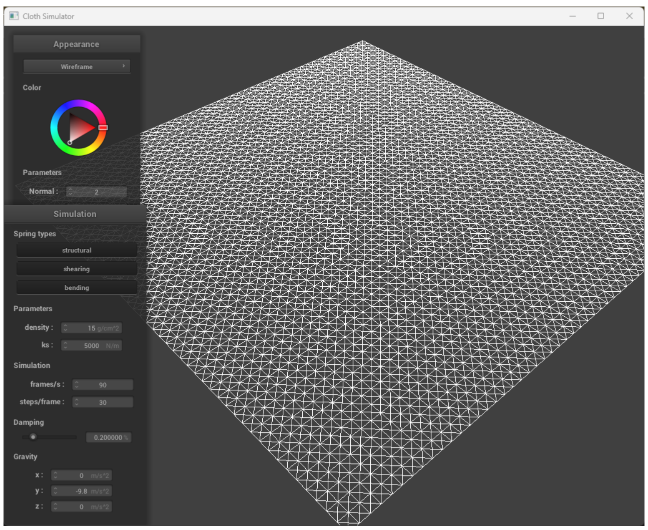
Project 4: Cloth Simulator

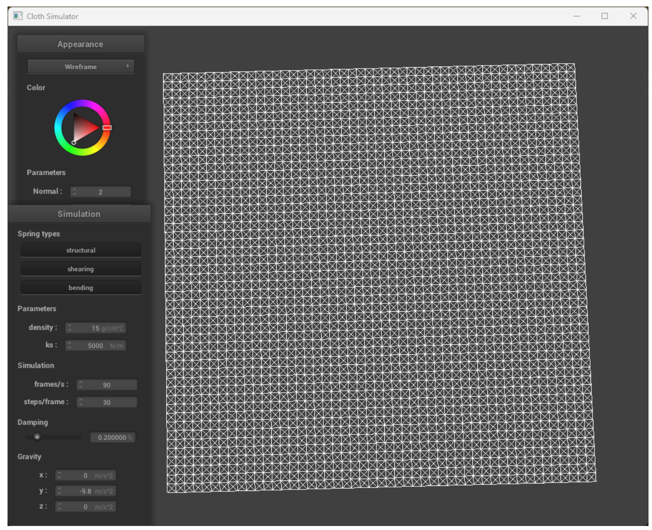
Overview

In this project, we built a cloth simulator using the mass-spring system that supports self-collision, object-collision, and shaders. This project is really cool.

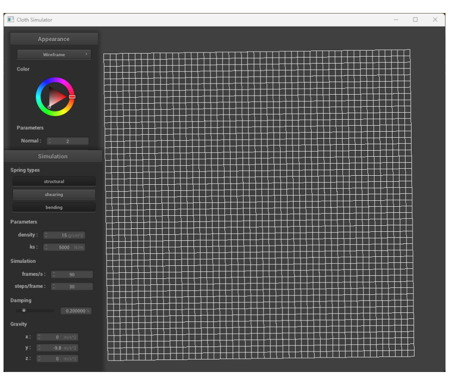
Part I: Masses and Springs

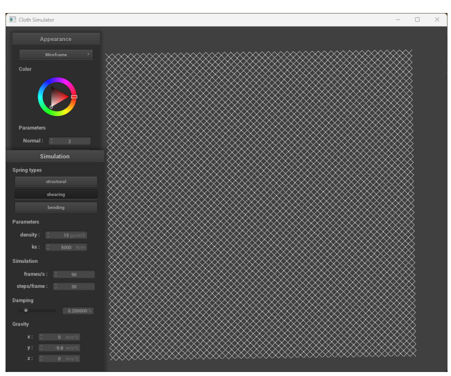
Take some screenshots of scene/pinned2.json from a viewing angle where you can clearly see the cloth wireframe to show the structure of your point masses and springs.

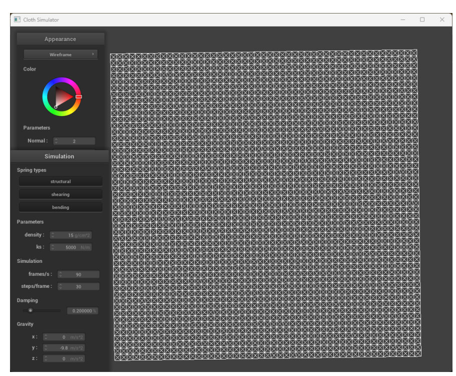




Show us what the wireframe looks like (1) without any shearing constraints, (2) with only shearing constraints, and (3) with all constraints.


without shearing constraints

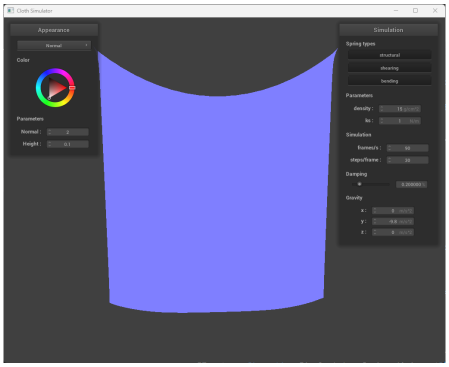

with only shearing constraint


with all constraints

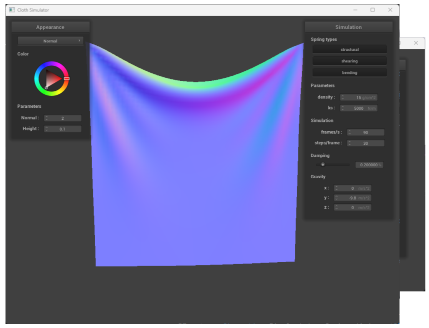
Part II: Simulation via numerical integration

Describe the effects of changing the spring constant ks; how does the cloth behave from start to rest with a very low ks? A high ks?

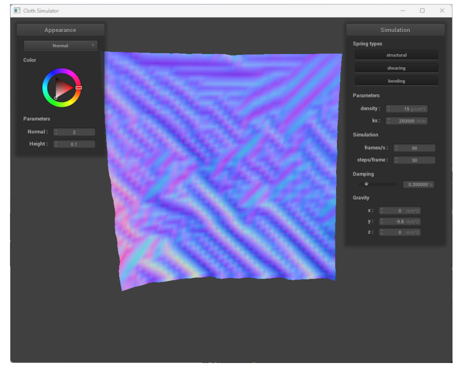
Ks determines the stiffness of the springs. A really low ks makes the cloth very loose and the cloth sags a lot. A high ks makes the cloth very stiff, and as a result, more resistant to external forces. Visually, the cloth is more "crisp" with a high ks.



Small ks



Default ks

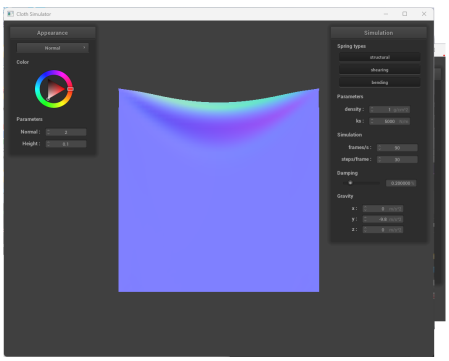


Big ks

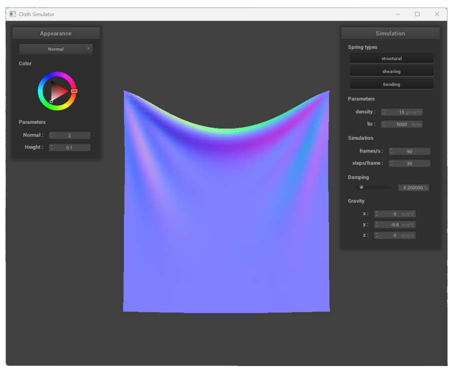
With big ks, the cloth seems to be extremely tight that it tries to pull itself together. This reminds me balloons.

What about for density?

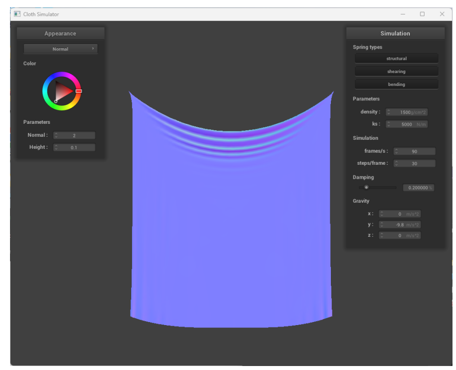
Density changes the cloth's mass per unit and therefore, inertia and resistance to external forces. A higher density makes the cloth heavier and more resistant to external forces, making it harder to move around; the cloth also tend to sag more. A lower density makes the cloth lighter and more susceptible to external forces, making it easier to move around; the cloth also tends to sag less.



Low density



Default density

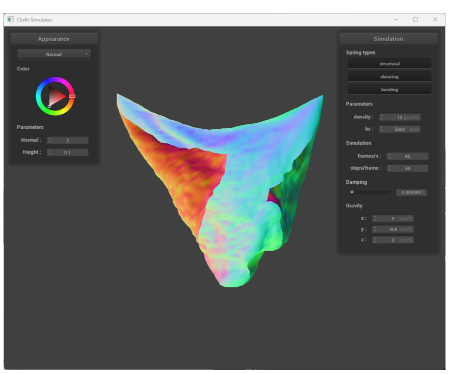


High density

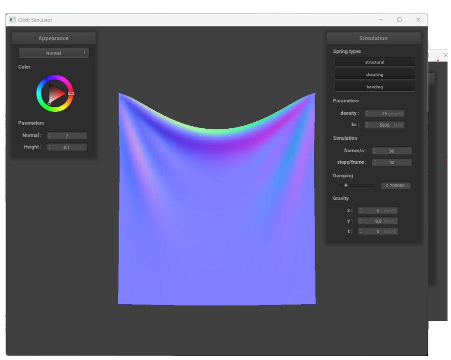
With low density, the cloth appears to be visually lighter too since it sags much less. It makes the viewer feel as if less force is pulling the cloth down.

What about for damping?

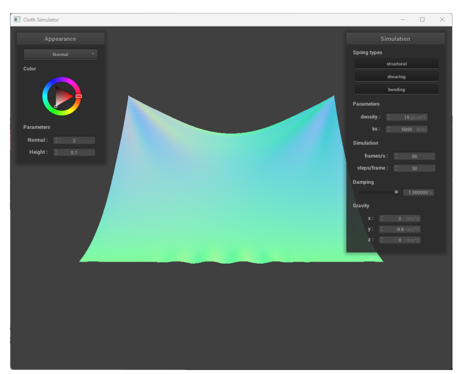
Damping determines the amount of energy loss for each simulation. A higher damping makes the cloth lose energy faster, making it less likely to oscillate. A lower damping makes the cloth lose energy slower, making it more likely to oscillate, i.e. more dynamic yet less stable.



Low damping



Default damping



High damping

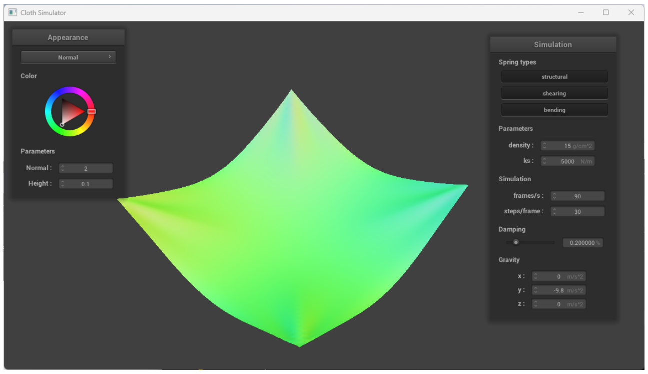
The images above are taken 1 second into the simulation. With damping set to zero, the cloth oscillates a lot; with higher damping values, the cloth oscillates a lot less and stops really quickly, with gravity being the only force that pulls the cloth down.

For each of the above, observe any noticeable differences in the cloth compared to the default parameters and show us some screenshots of those interesting differences and describe when they occur.

see above.

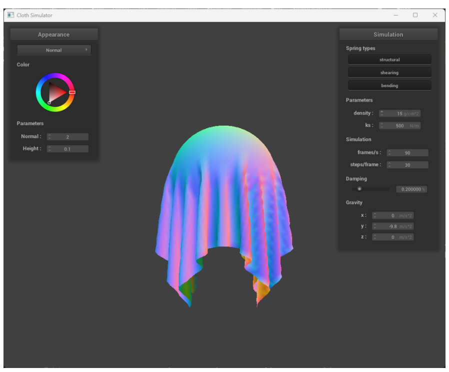
Show us a screenshot of your shaded cloth from scene/pinned4.json in its final resting state! If you choose to use different parameters than the default ones, please list them.

Final state of pinned4.json

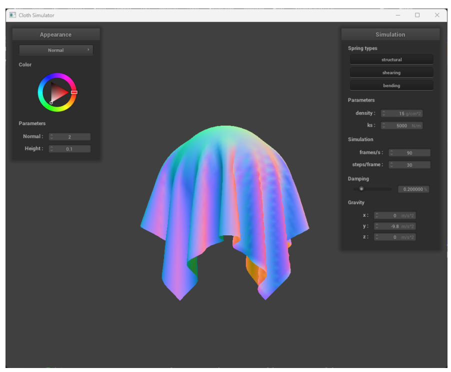


Part III: Handling collisions with other objects

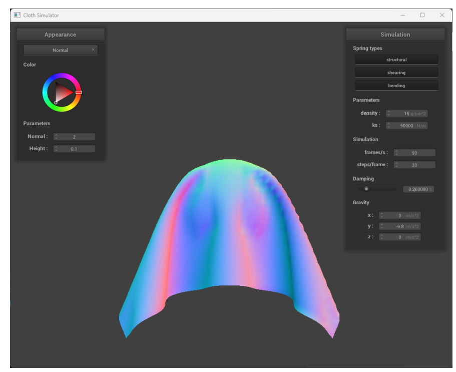
Show us screenshots of your shaded cloth from scene/sphere.json in its final resting state on the sphere using the default ks = 5000 as well as with ks = 500 and ks = 50000. Describe the differences in the results.



ks = 500



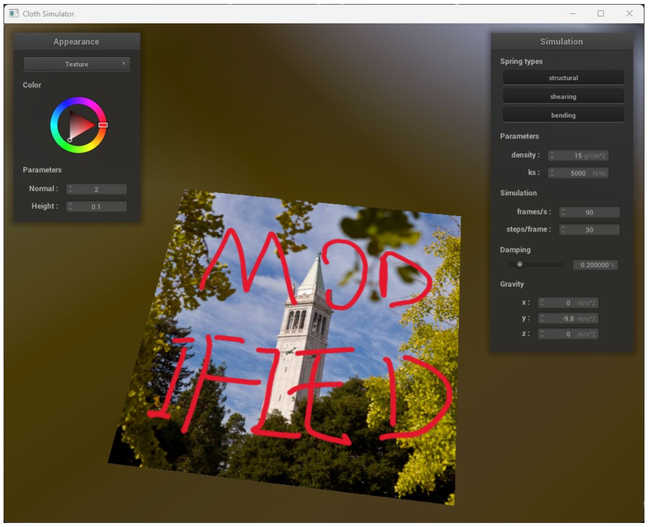
ks = 5000



ks = 50000

Simulations with lower ks appears to be less stiff and more soggy. The cloth drapes more loosely on the sphere. Simulations with high ks(50000) looks very stiff and crisp, and the cloth drapes very tightly on the sphere, with most part "folding" together on one of the 4 sides, with much fewer folds.

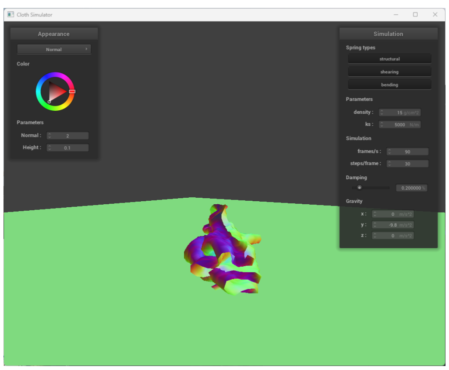
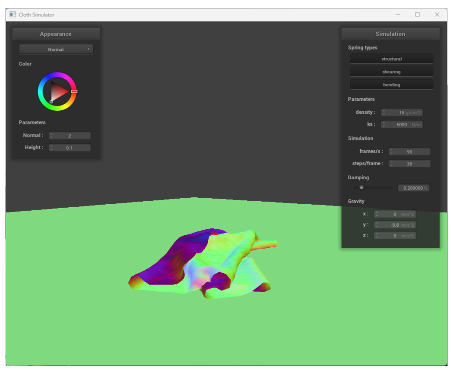
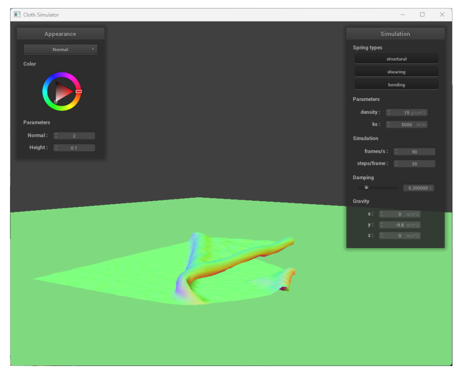
Show us a screenshot of your shaded cloth lying peacefully at rest on the plane. If you haven't by now, feel free to express your colorful creativity with the cloth! (You will need to complete the shaders portion first to show custom colors.)



peace

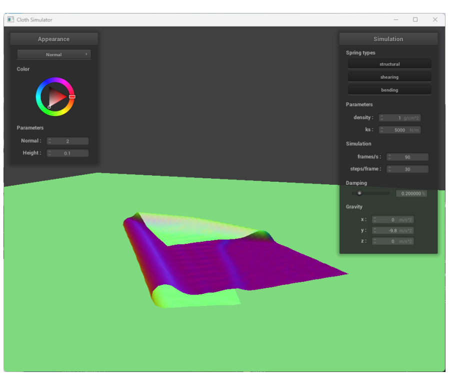
Part IV: Handling self-collisions

Show us at least 3 screenshots that document how your cloth falls and folds on itself, starting with an early, initial self-collision and ending with the cloth at a more restful state (even if it is still slightly bouncy on the ground).

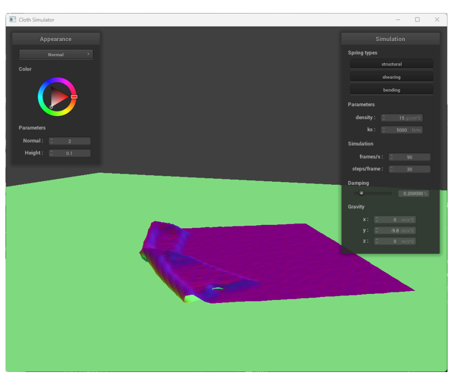
  

Vary the density as well as ks and describe with words and screenshots how they affect the behavior of the cloth as it falls on itself.

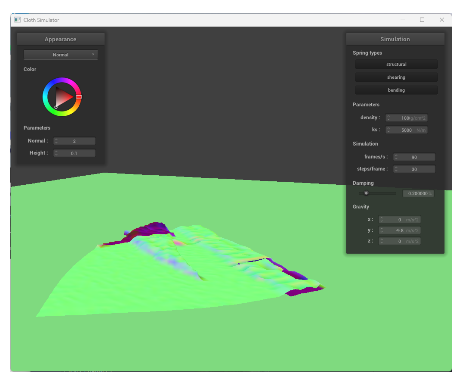
Higher density makes the cloth heavier, and it tends to collapse onto itself more, lower density makes the cloth lighter and it tends to spread out more. The fold is also more pronounced with higher density.



low density

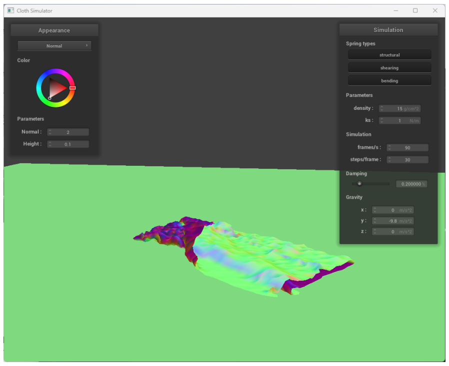


default density

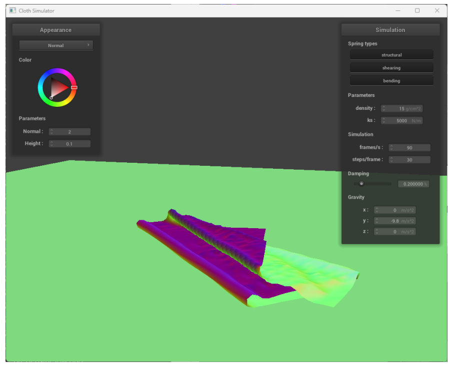


high density

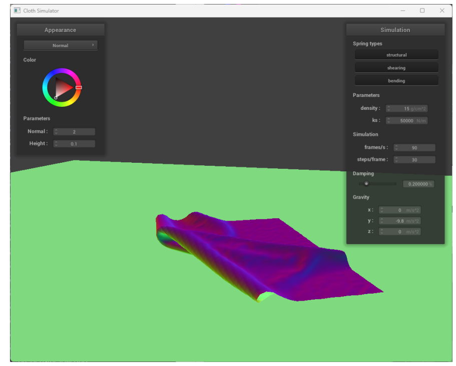
Variation in ks affects the speed of the cloth's collapse. Higher ks makes the cloth collapse faster, lower ks makes the cloth collapse slower, as the cloth has less force to pull itself together. The following 3 images are taken 5 seconds into the simulation; cloth with higher ks appears to be "folding" more quickly and looks more "crisp" due to the more pronounced folds as a result of the higher ks. Cloth with low ks looks as if it's having a hard time pulling itself together.



low ks



default ks



high ks

Part V: Shaders

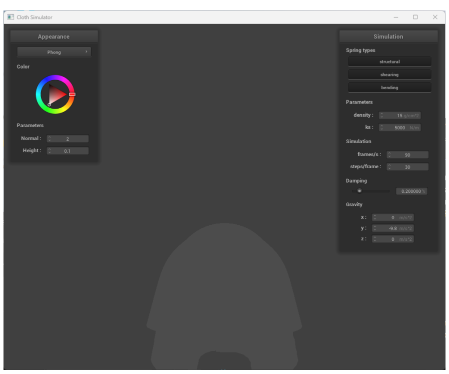
Explain in your own words what is a shader program and how vertex and fragment shaders work together to create lighting and material effects.

A shader program is a program, typically run by GPU, that takes in a vertex or a fragment and a bunch of parameters, and outputs how the vertex or fragment should be rendered.

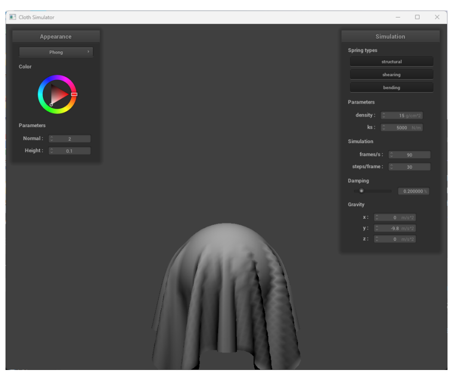
Vertex and fragment shaders are the two main components of the shader program. Vertex shaders process individual vertices of a 3D mesh, outputting their position and shading properties such as color and texture. Fragment shaders process fragments of vertex shader's outputs, for example, pixels, and output the final color of the pixel, which is then rendered on the screen.

Explain the Blinn-Phong shading model in your own words. Show a screenshot of your Blinn-Phong shader outputting only the ambient component, a screen shot only outputting the diffuse component, a screen shot only outputting the specular component, and one using the entire Blinn-Phong model.

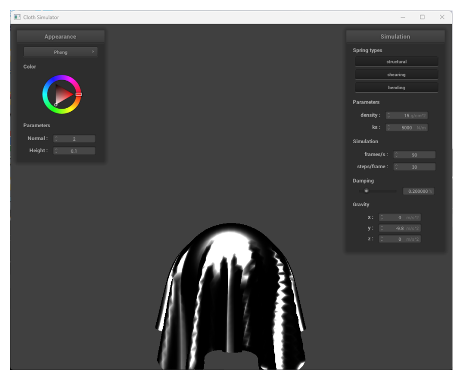
The Blinn-Phong shading model is a superset of diffuse shading. Blinn-Phong shading comprises 3 parts: diffuse lighting, ambient lighting, and specular lighting. Diffuse lighting is the most basic reflection from an object, while specular lighting depends on the camera angle, and ambient lighting is the light that is reflected from the environment. Together, the 3 lighting components creates a realistic rendering of the object.



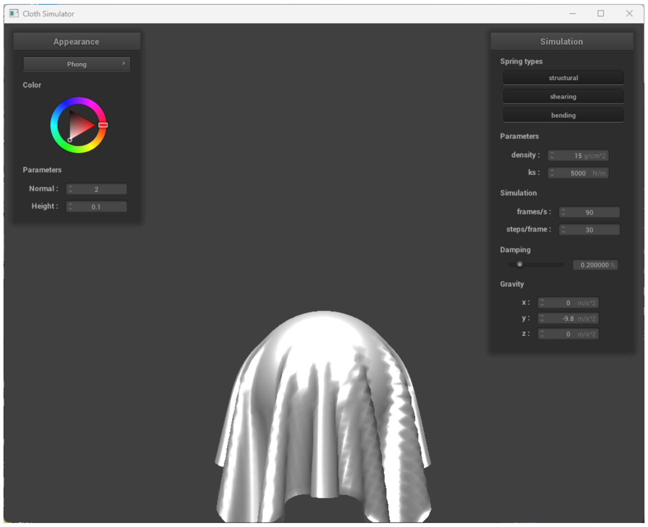
Only ambient



Only diffuse

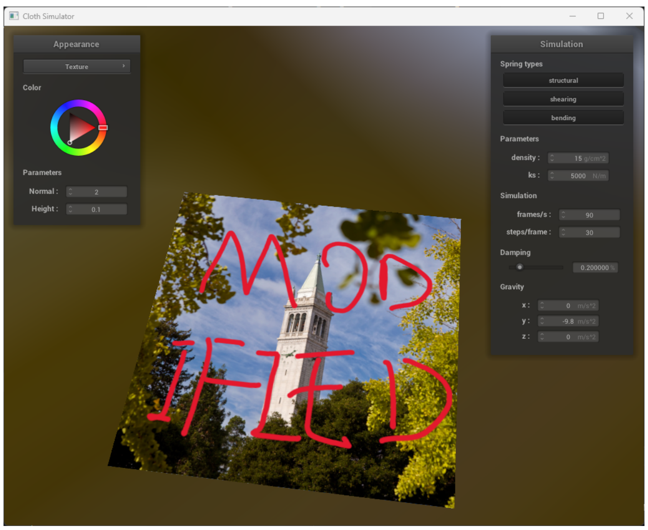


Only specular



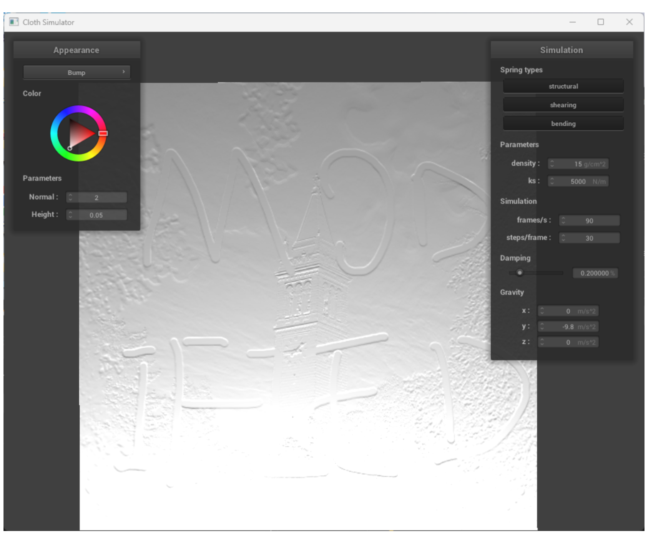
Ambient, diffuse, and specular

Show a screenshot of your texture mapping shader using your own custom texture by modifying the textures in /textures/.

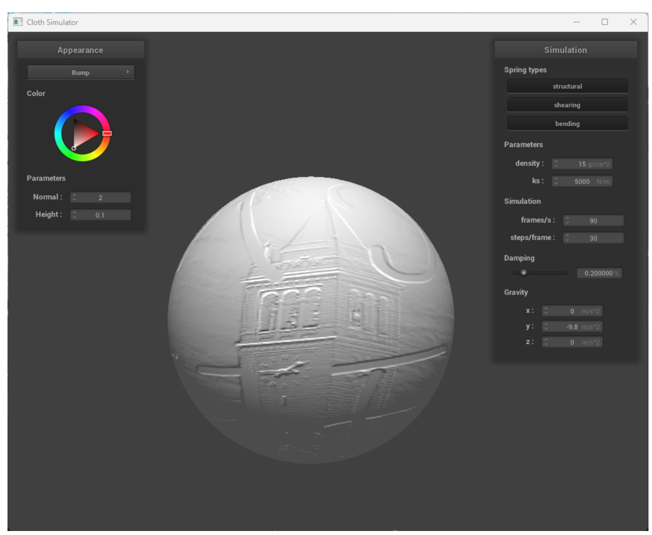


Modified texture

Show a screenshot of bump mapping on the cloth and on the sphere.

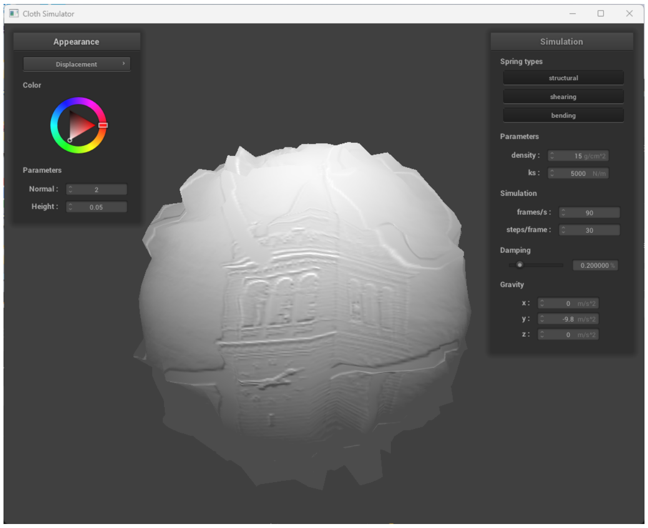


Bump mapping on cloth



Bump mapping on sphere

Show a screenshot of displacement mapping on the sphere. Use the same texture for both renders. You can either provide your own texture or use one of the ones in the textures directory, BUT choose one that's not the default texture_2.png.

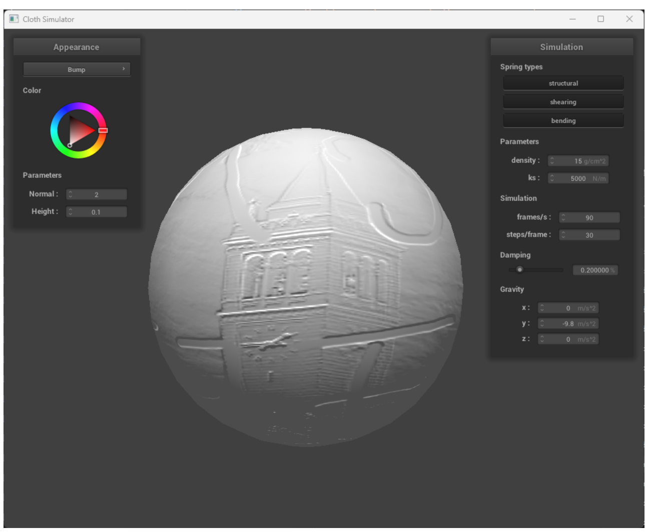


Displacement mapping on sphere

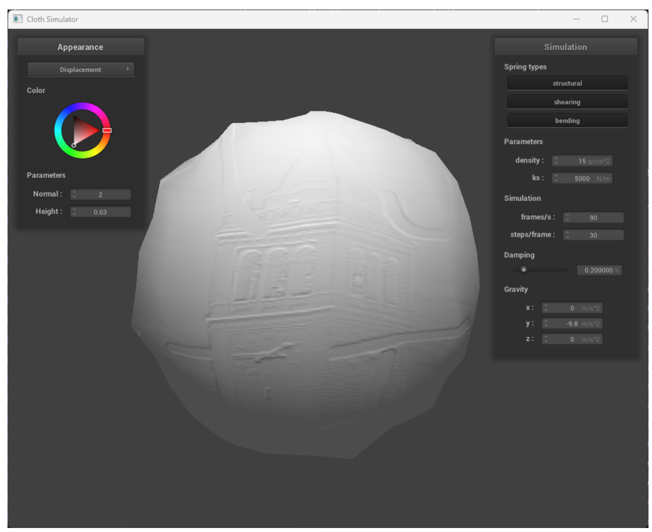
Compare the two approaches and resulting renders in your own words.

While both approaches give the illusion of a 3D object, bump mapping provides a more stable and less "3D" appearance than displacement mapping. Displacement mapping is more "3D", but may look weird depending on the depth of the displacement, as it directly changes vertex positions. Bump mapping is more stable, but may look less "3D" depending on the strength of the bump. Both react to light.

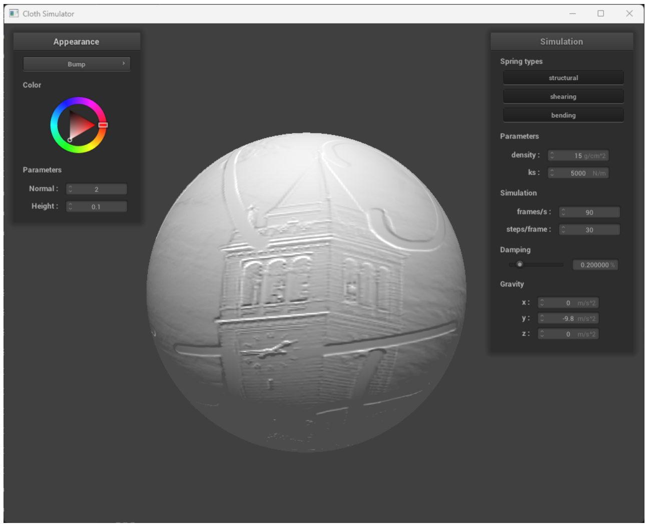
Compare how your the two shaders react to the sphere by changing the sphere mesh's coarseness by using -o 16 -a 16 and then -o 128 -a 128.



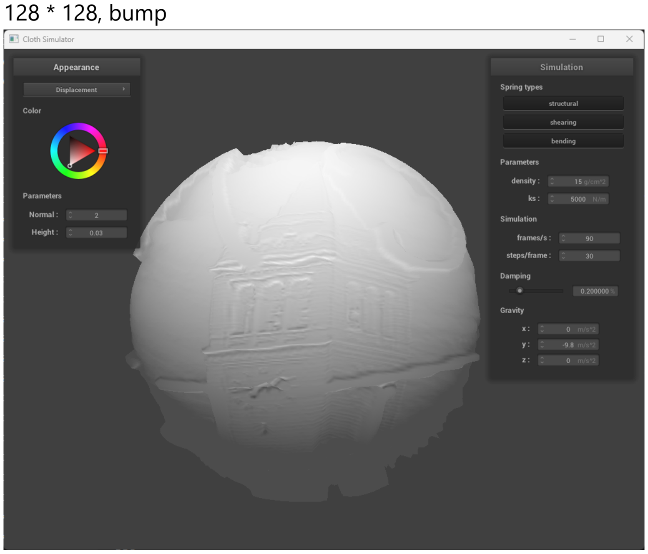
16 * 16, bump



16 * 16, displacement



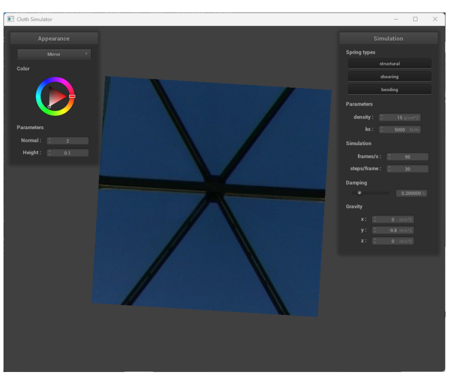
128 * 128, bump



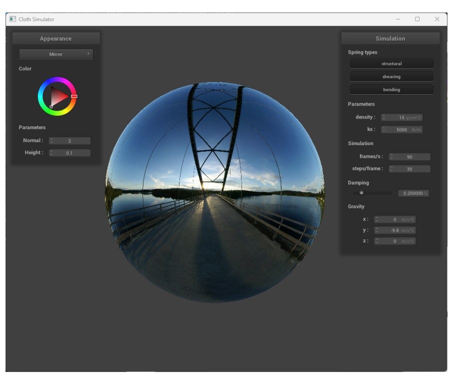
128 * 128, displacement

Per the above comparison, changes in coarseness manifest most clearly on the displacement shader. The higher resolution the sphere has, the more protrusion i.e. 3D-ness the displacement shader shows. The bump shader, on the other hand, is more stable and less affected by coarseness. If one look very closely, the bump shader does show some extra details on the higher resolution sphere, but it is not as pronounced as the displacement shader.

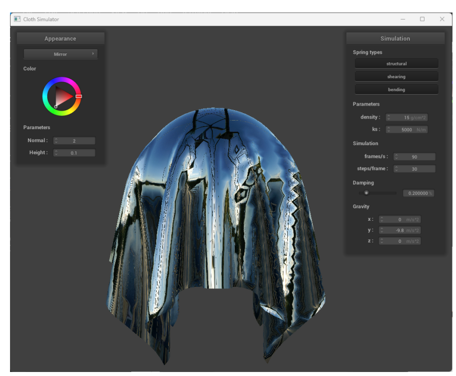
Show a screenshot of your mirror shader on the cloth and on the sphere.



Cloth



Sphere



Cloth on the sphere