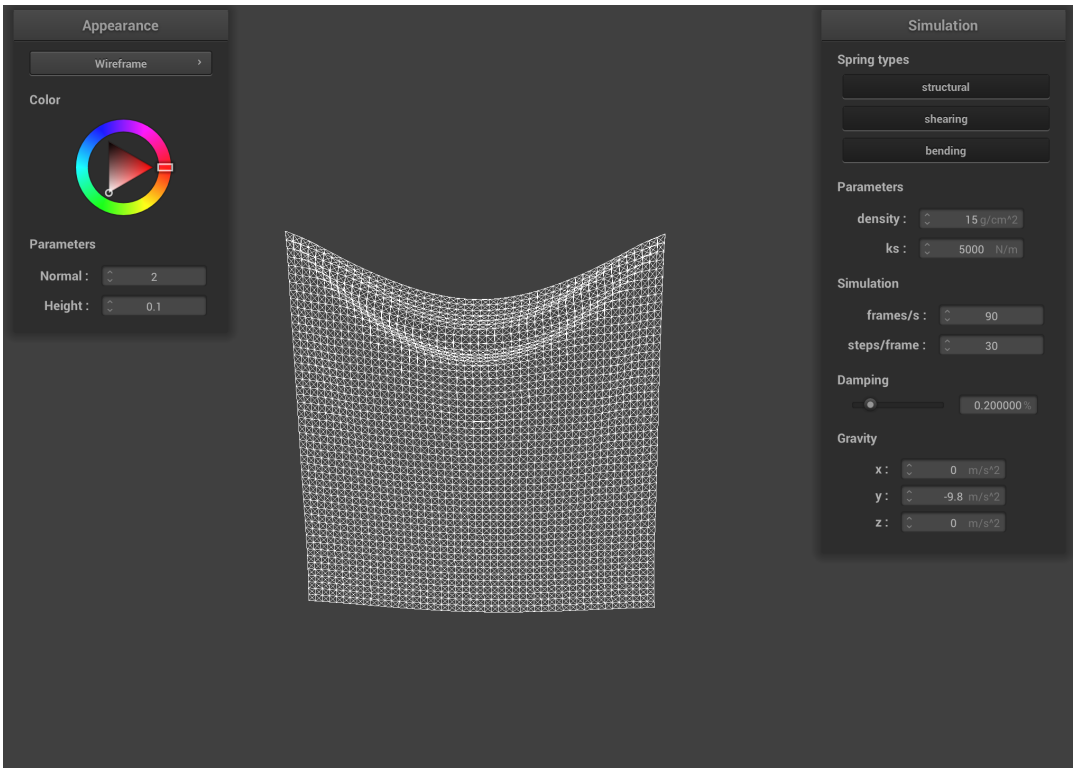
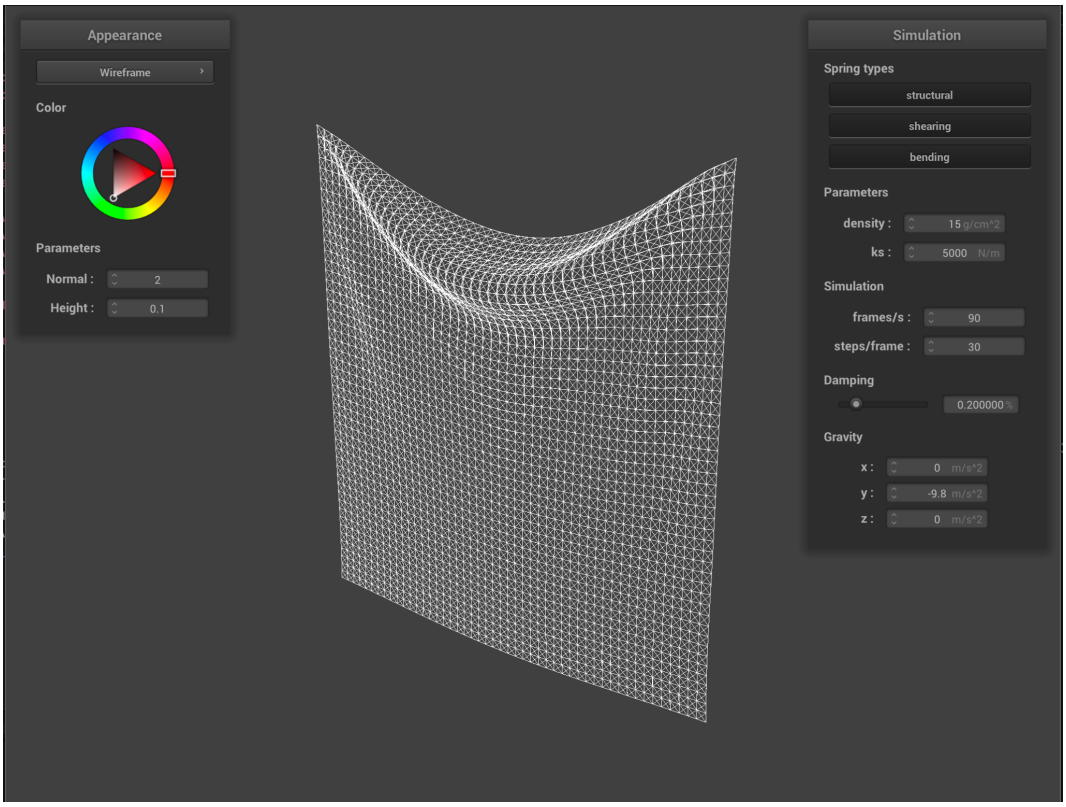
Overview

- In this project, we simulated many aspects of cloth as well as rendering them. First, we created the cloth itself using massPoints and springs, then we simulated how the cloth would move using both external forces and spring correction forces. Afterwards, we handled the collisions of the cloth and both spheres and planes by using offsets when they collided. Next, we needed to make sure the the cloth would collide with itself by using a hashmap to check for collisions. Lastly, we used a lot of different shaders for our cloth, including diffuse, blinn-phong, texture, bump/displacement, and mirror. We didn't realize the spec had specific parameters and spent a long time trying to figure out what was wrong.

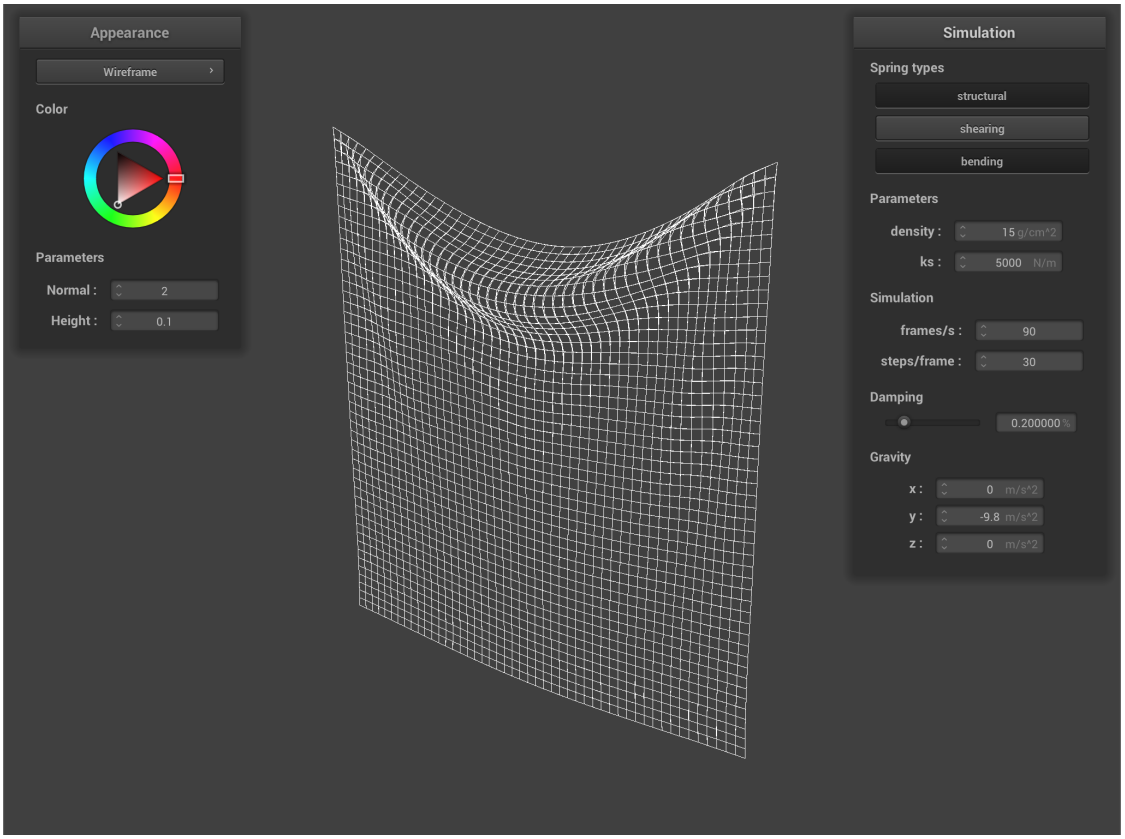
Part 1

- For part 1, we created the cloth itself using massPoints and springs, with an evenly spaced grid with num width points and num height points. Then we applied structual, shear, and bending constraints between point masses.
- 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.
  - 

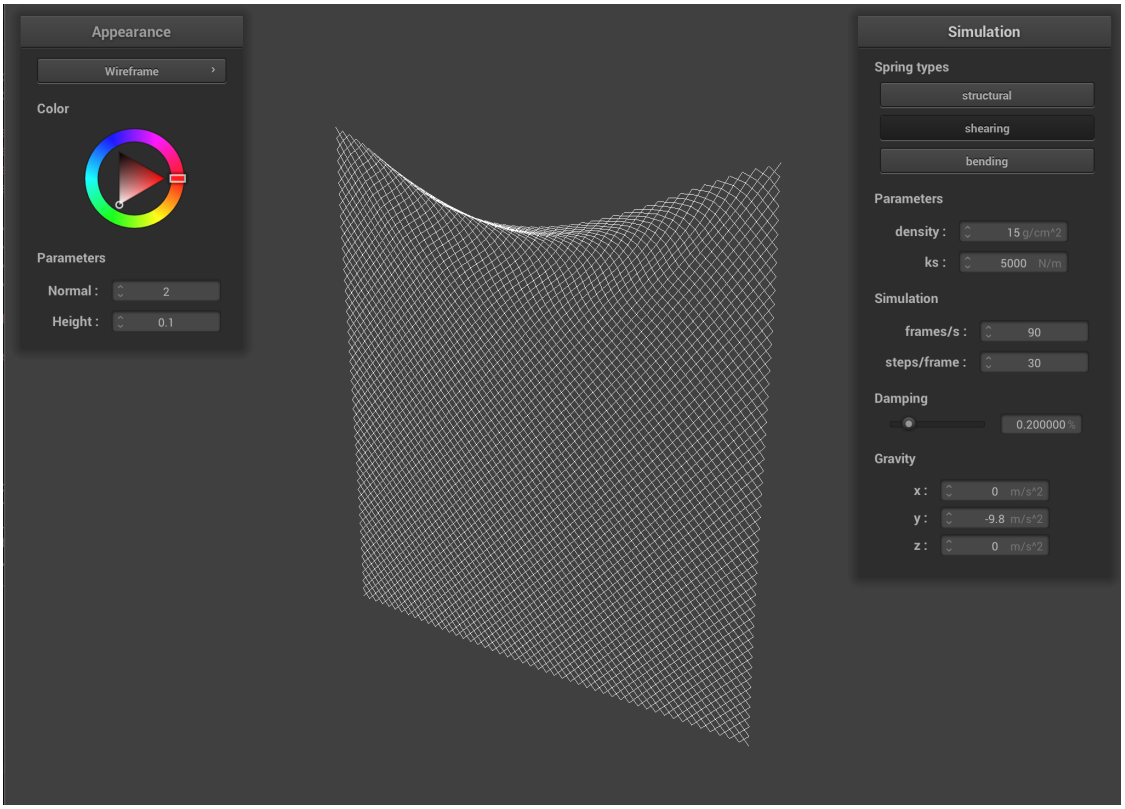
- 

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

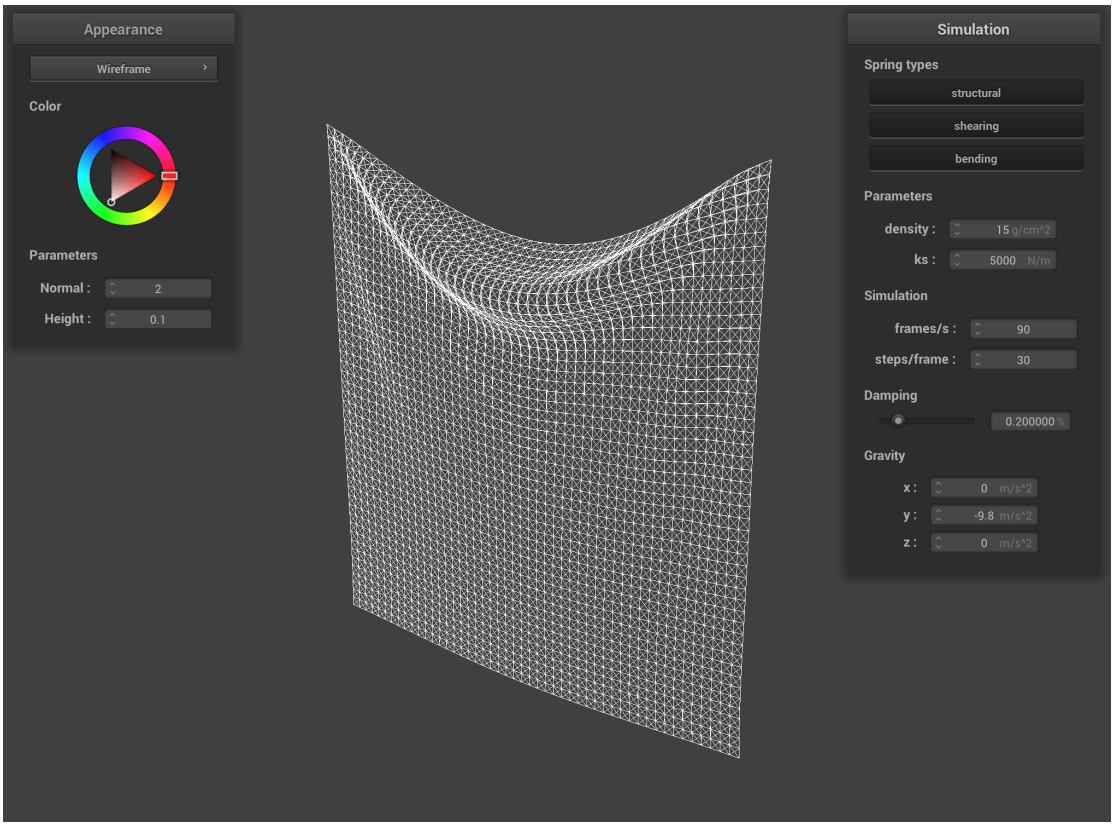
- 1.


Appearance
Wireframe
Color
Parameters
Normal :
2
Height :
0.1
Simulation
Spring types
structural
shearing
bending
Parameters
density :
15 g/cm^2
ks :
5000 N/m
Simulation
frames/s :
90
steps/frame :
30
Damping
0.200000 %
Gravity
x :
0 m/s^2
y :
-9.8 m/s^2
z :
0 m/s^2

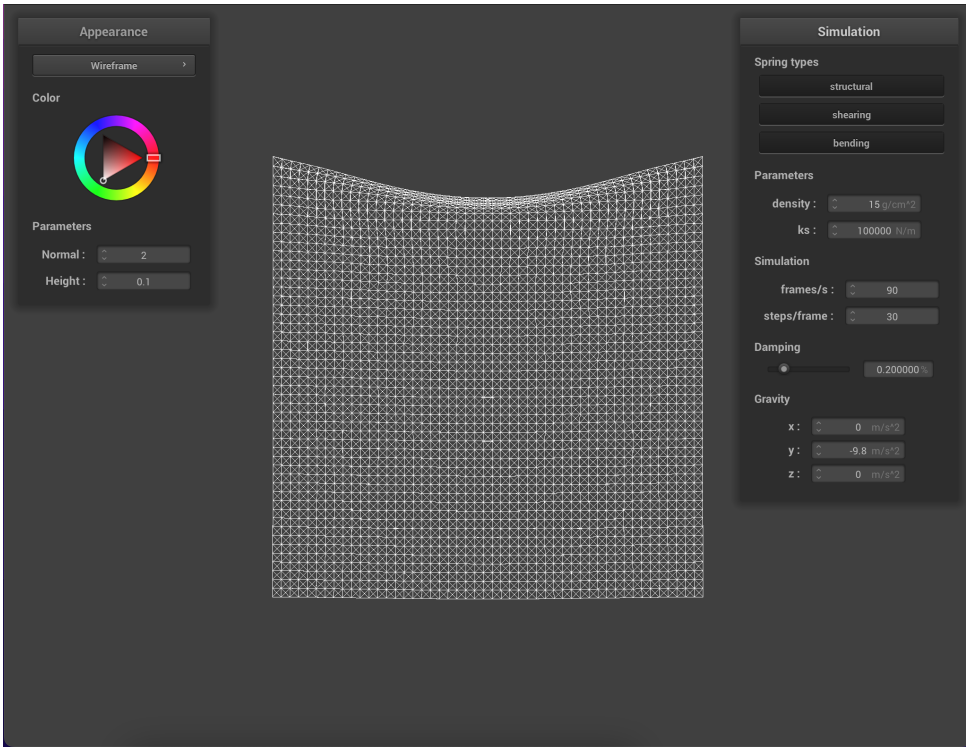
- 2.

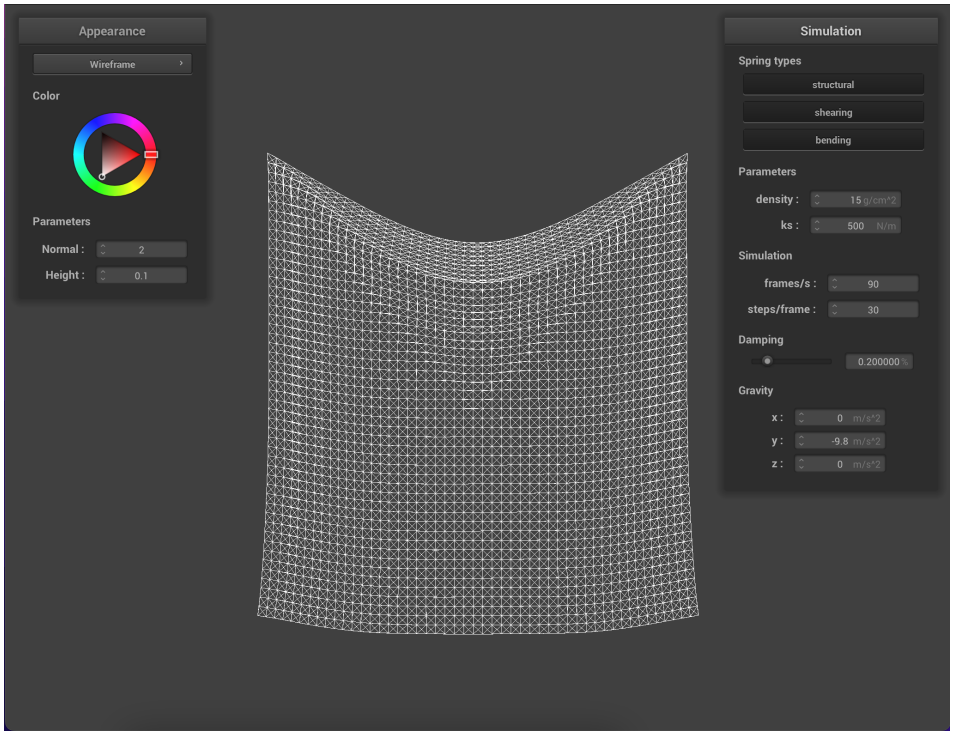

Appearance
Wireframe
Color
Parameters
Normal : 2
Height : 0.1
Simulation
Spring types
structural
shearing
bending
Parameters
density : 15 g/cm^2
ks : 5000 N/m
Simulation
frames/s : 90
steps/frame : 30
Damping
0.200000 %
Gravity
x : 0 m/s^2
y : -9.8 m/s^2
z : 0 m/s^2

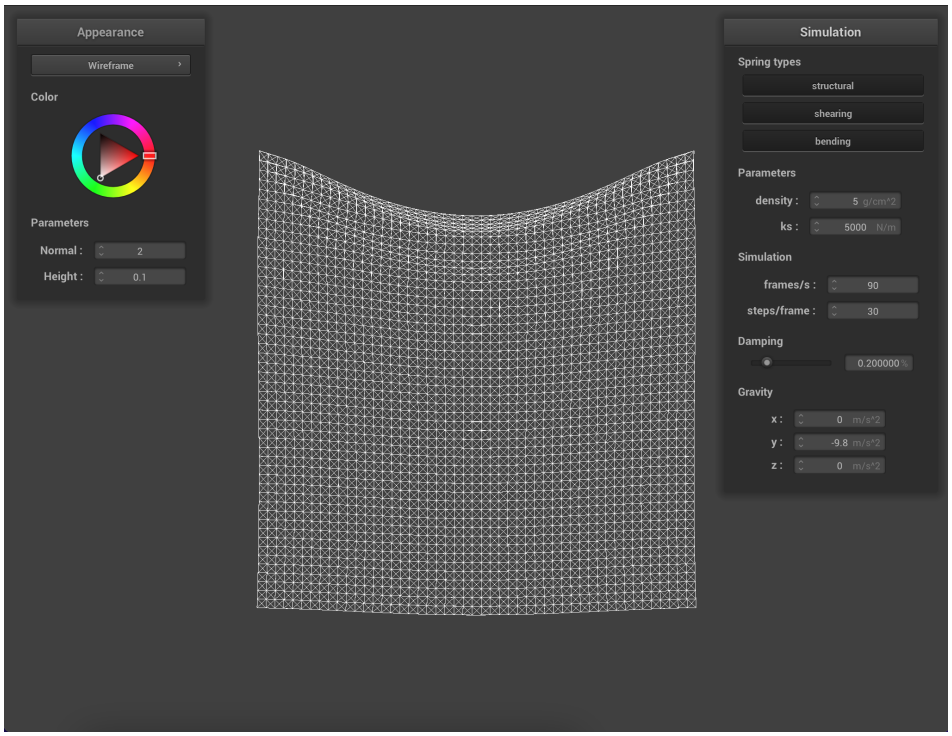
- 3.


Appearance
Wireframe
Color
Parameters
Normal : 2
Height : 0.1
Simulation
Spring types
structural
shearing
bending
Parameters
density : 15 g/cm^2
ks : 5000 N/m
Simulation
frames/s : 90
steps/frame : 30
Damping
0.200000 %
Gravity
x : 0 m/s^2
y : -9.8 m/s^2
z : 0 m/s^2

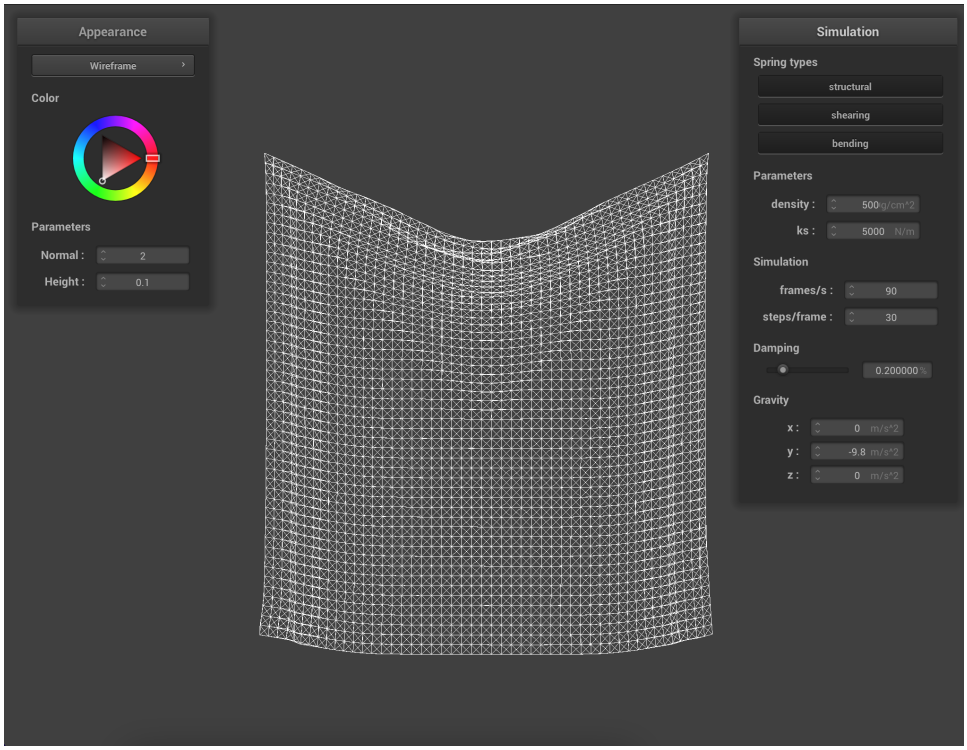
Part 2

- For part 2, we computed the total force acting on each point mass by changing the vector forces that each point mass have. We used the equation F = ks * (||pa - pb|| - l). Then we used verlet integration to compute new point mass positions. We computed the point mass's new position in each time t + dt and added some damping for friction. Lastly, we corrected the two point masses' positions such that the spring's length is at most 10% greater than its rest_length at the end of any time step for each spring through checking if they need to be adjusted/are pinned.
- Experiment with some the parameters in the simulation. To do so, pause the simulation at the start with `P`, modify the values of interest, and then resume by pressing `P` again. You can also restart the simulation at any time from the cloth's starting position by pressing `R`.
    - 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`?
    - What about for `density`?
    - What about for `damping`?
    - 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.
        - High ks
- 

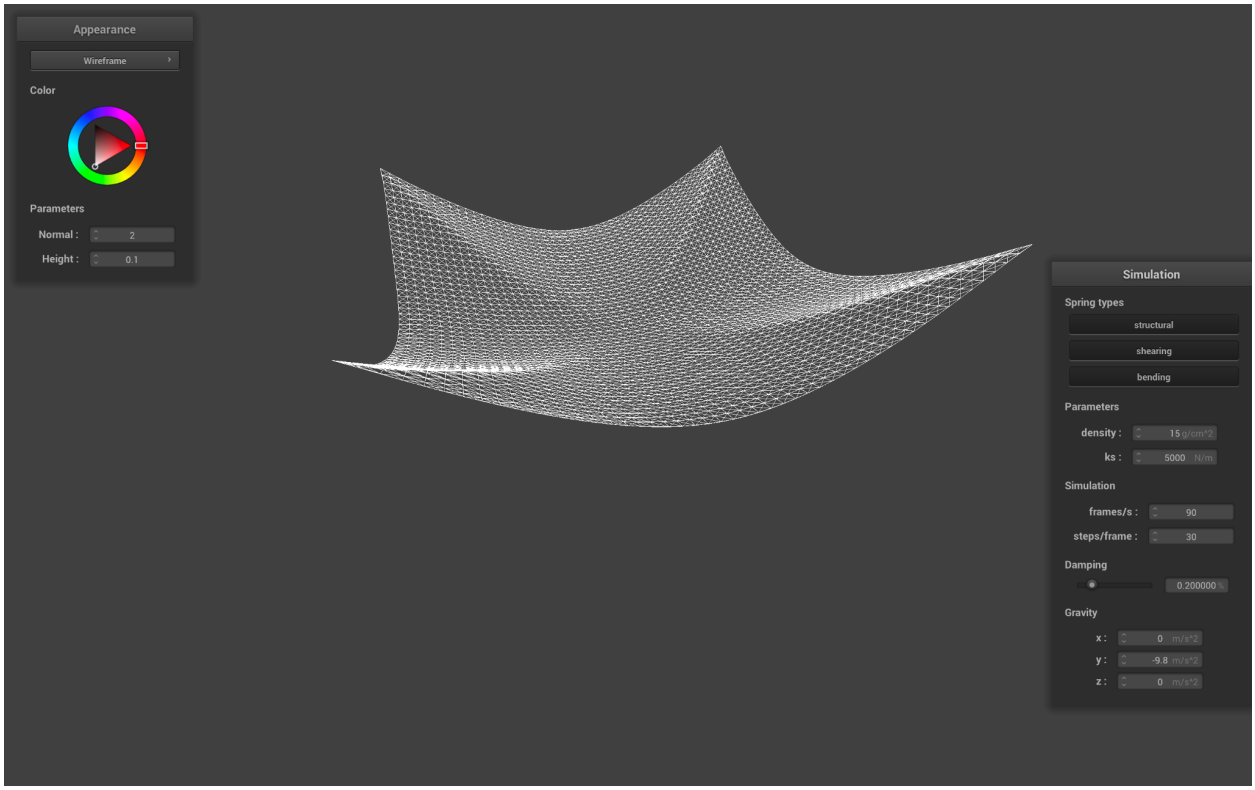
- Low ks
  - 

- High density
  - 

- Low density

- 

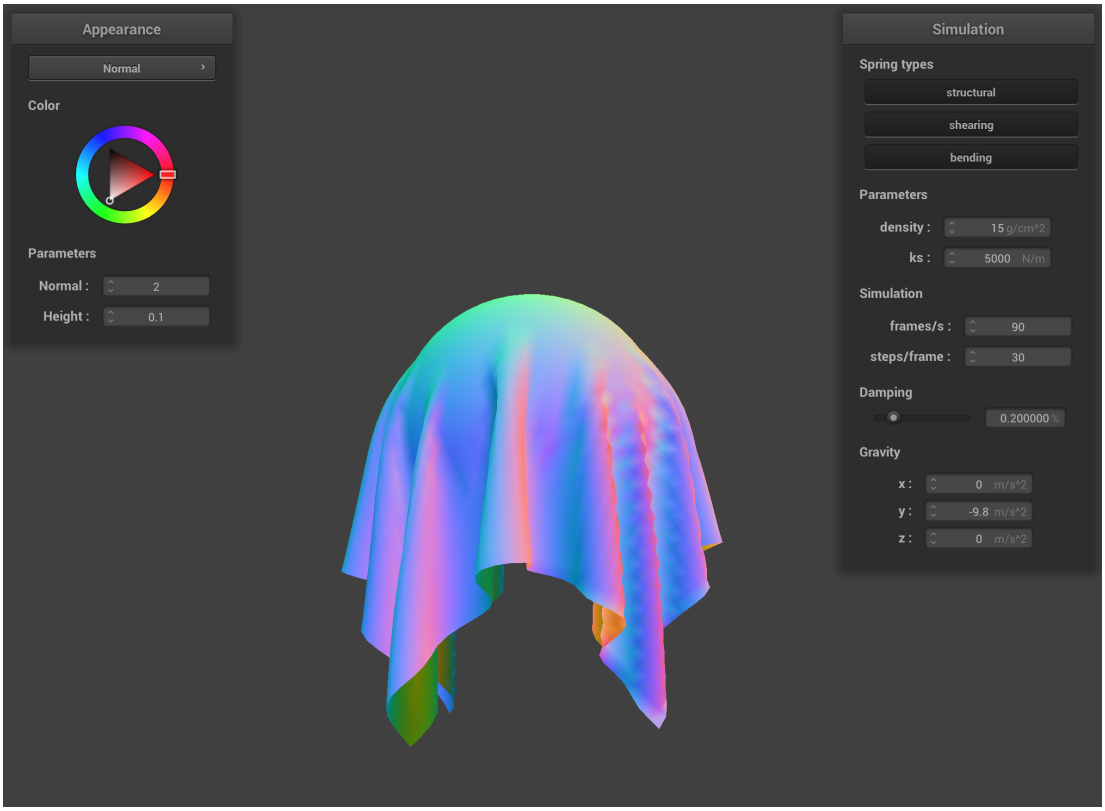
    - When there is more (high) density and lower ks, the cloth would seem to be dripped down more. Whereas with less (low) density and high ks, the cloth would seem to be tense.
    - When there is more damping, the cloth would end up at the same spot from the start to end. However, the less damping there is, the more the cloth swings around before getting to the ending state. With damping set to 0.95, the cloth would slowly and stiffly fall to its resting position. With damping set close to 0, the cloth would swing back and forth, slowly losing energy as it settled down.
- 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.

- 

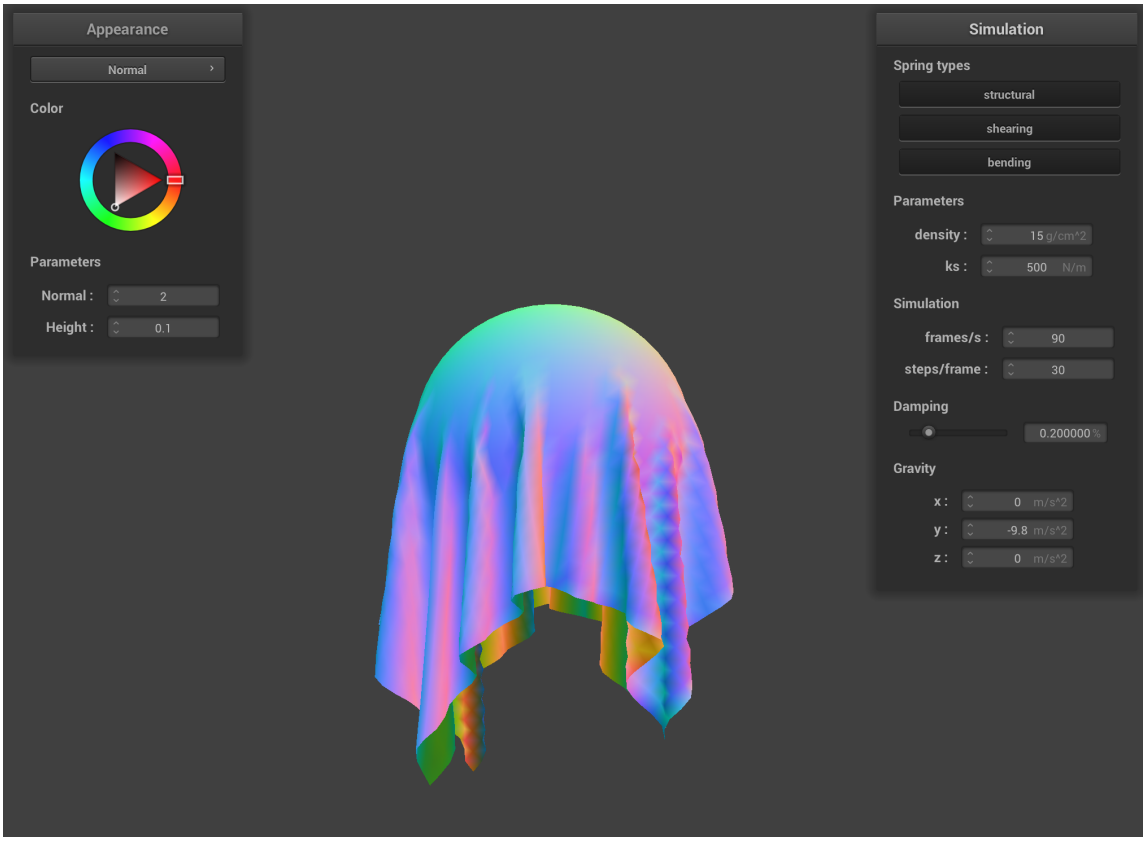
Part 3

- For part 3, we implemented solliding with sphere, which takes in a point mass and adjusts its position if it intersects with or is inside the sphere. If the point mass intersects with or is inside the sphere, then "bump" it up to the surface of the sphere. For planes, we calculate the tangent point and use it to see if we need to apply an offset.
- 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.

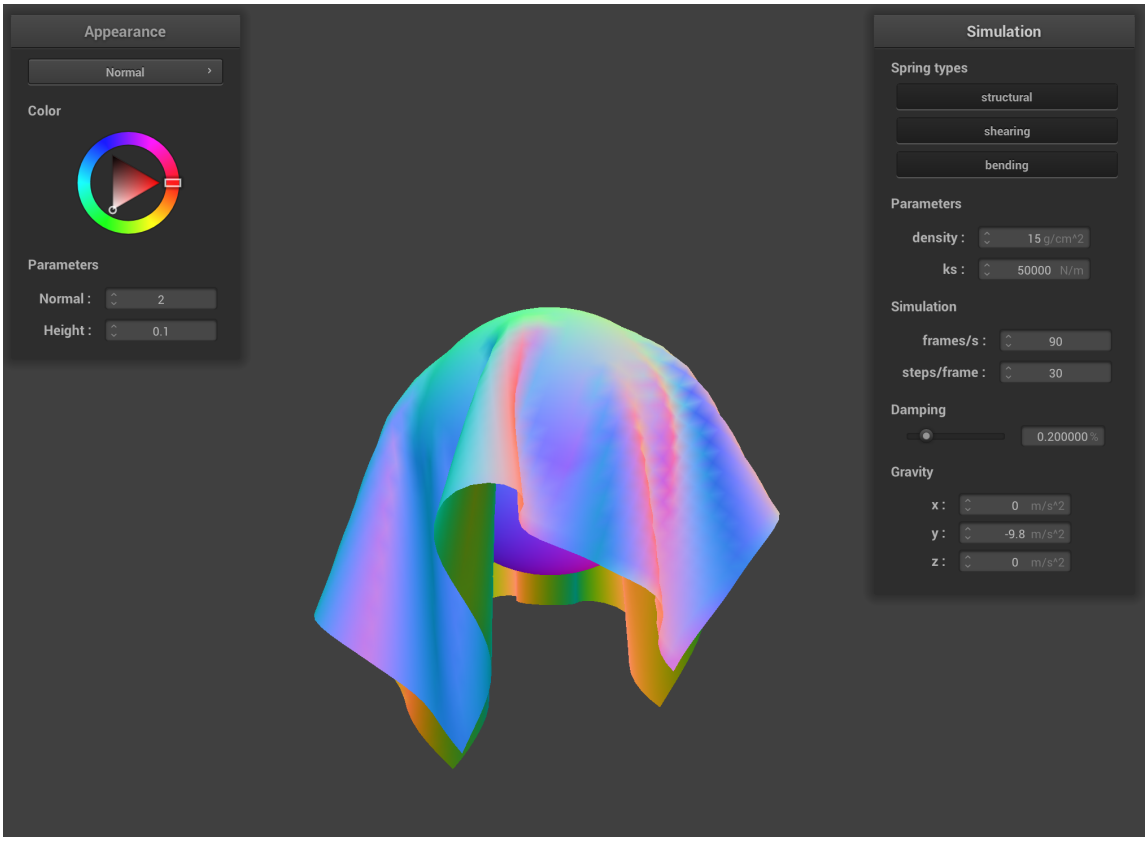
- 5000:


Appearance
Normal
Color
Parameters
Normal :
2
Height :
0.1
Simulation
Spring types
structural
shearing
bending
Parameters
density :
15 g/cm^2
ks :
5000 N/m
Simulation
frames/s :
90
steps/frame :
30
Damping
0.200000 %
Gravity
x :
0 m/s^2
y :
-9.8 m/s^2
z :
0 m/s^2

- 500:

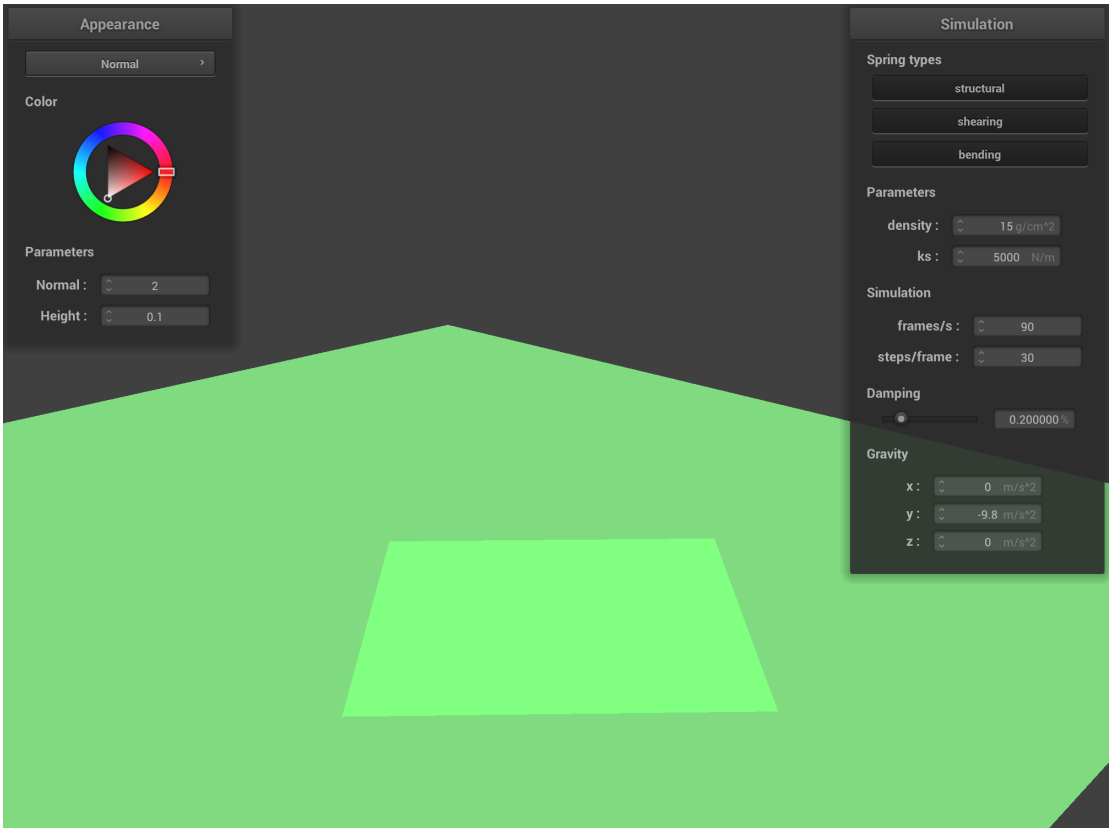
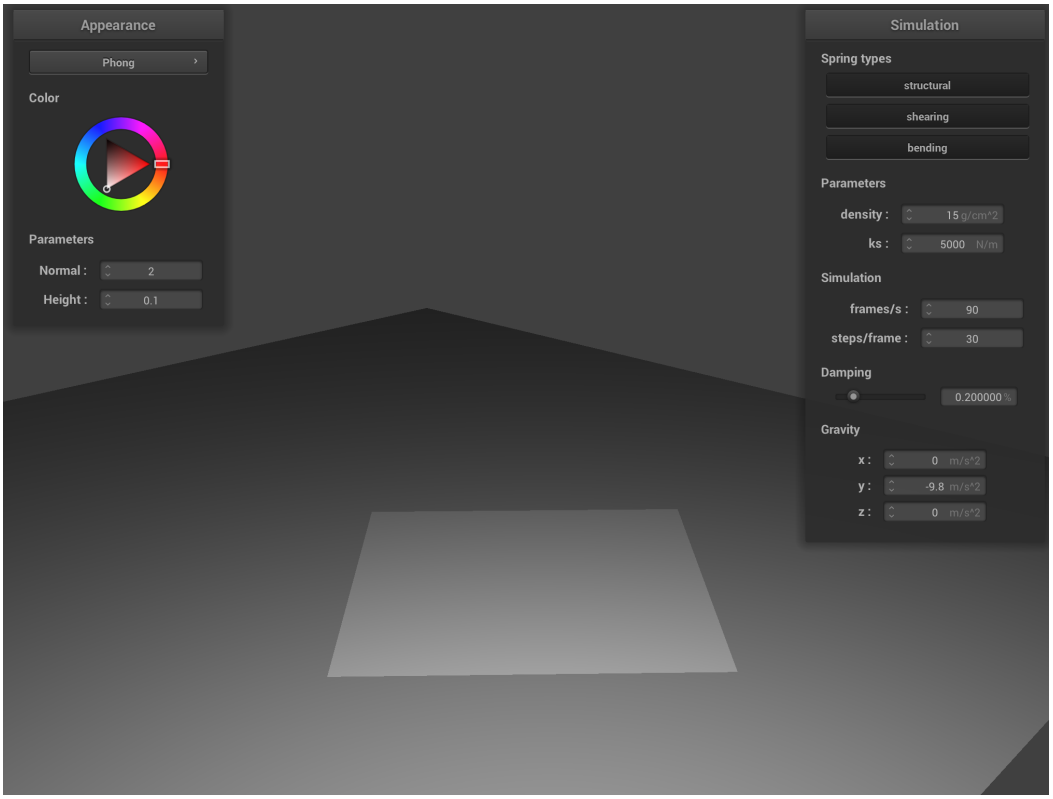

Appearance
Normal
Color
Parameters
Normal : 2
Height : 0.1
Simulation
Spring types
structural
shearing
bending
Parameters
density : 15 g/cm^2
ks : 500 N/m
Simulation
frames/s : 90
steps/frame : 30
Damping
0.200000 %
Gravity
x : 0 m/s^2
y : -9.8 m/s^2
z : 0 m/s^2

- 50000:

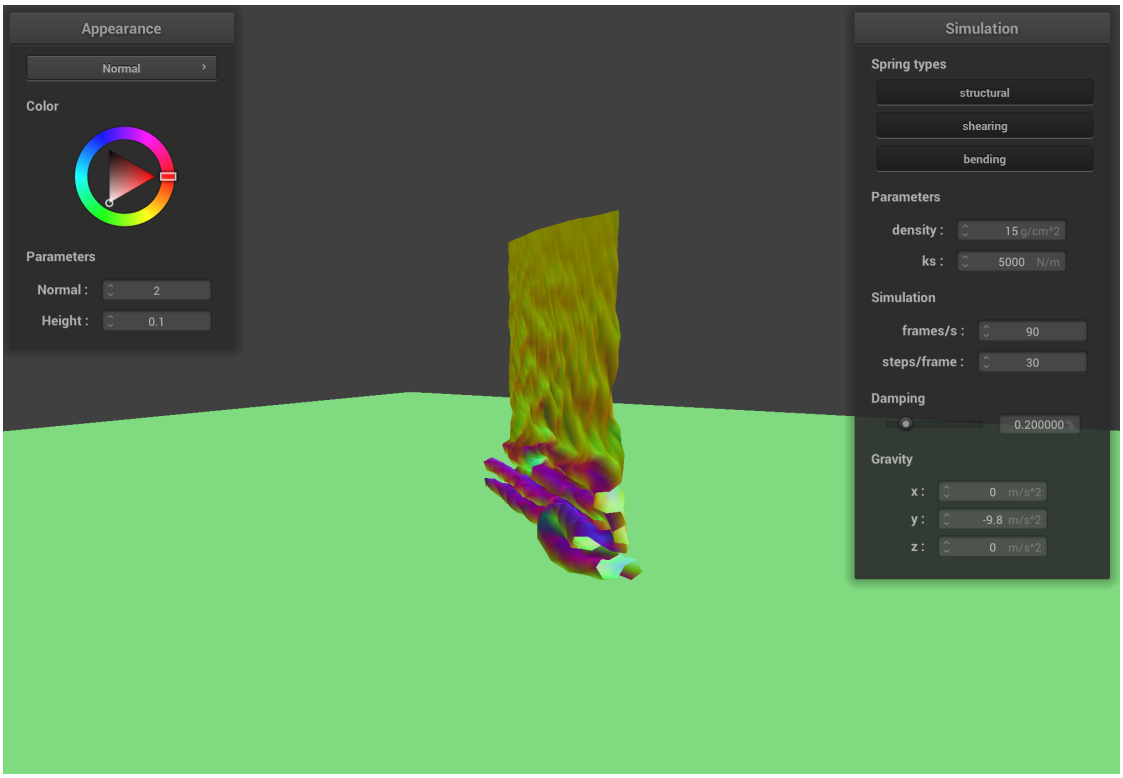


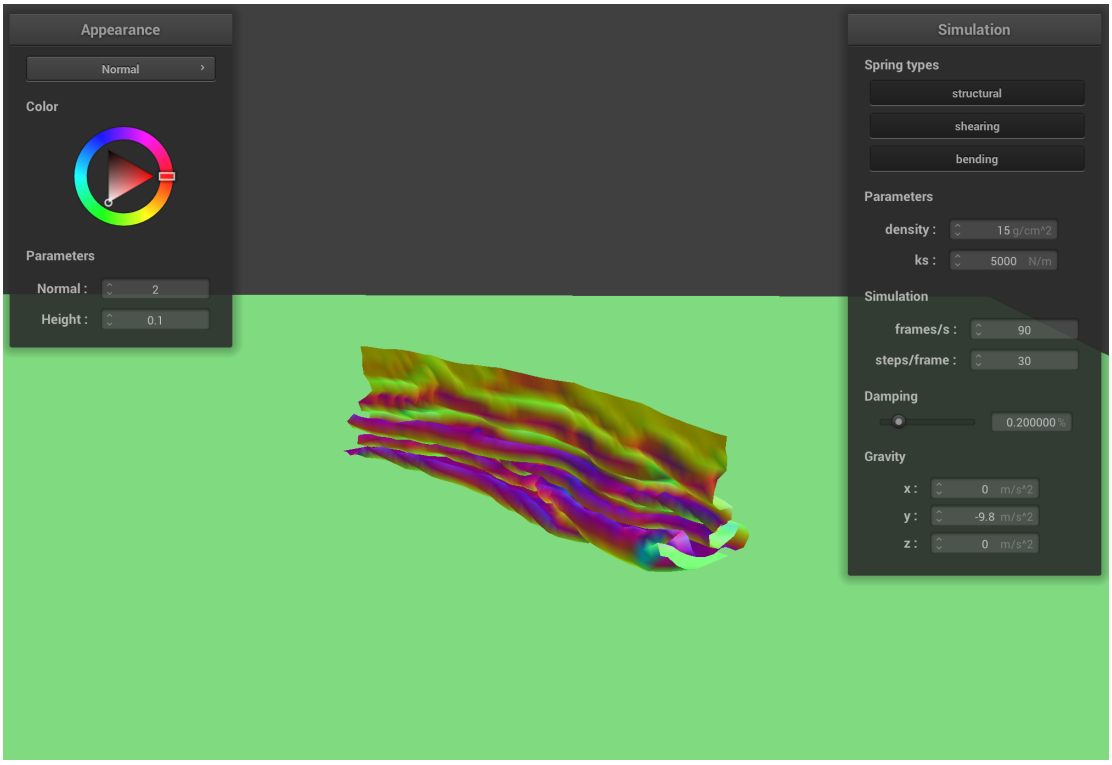
  - Differences: As ks increase, the cloth droops down less and less and as ks decreased, the cloth droops down lower and lowe

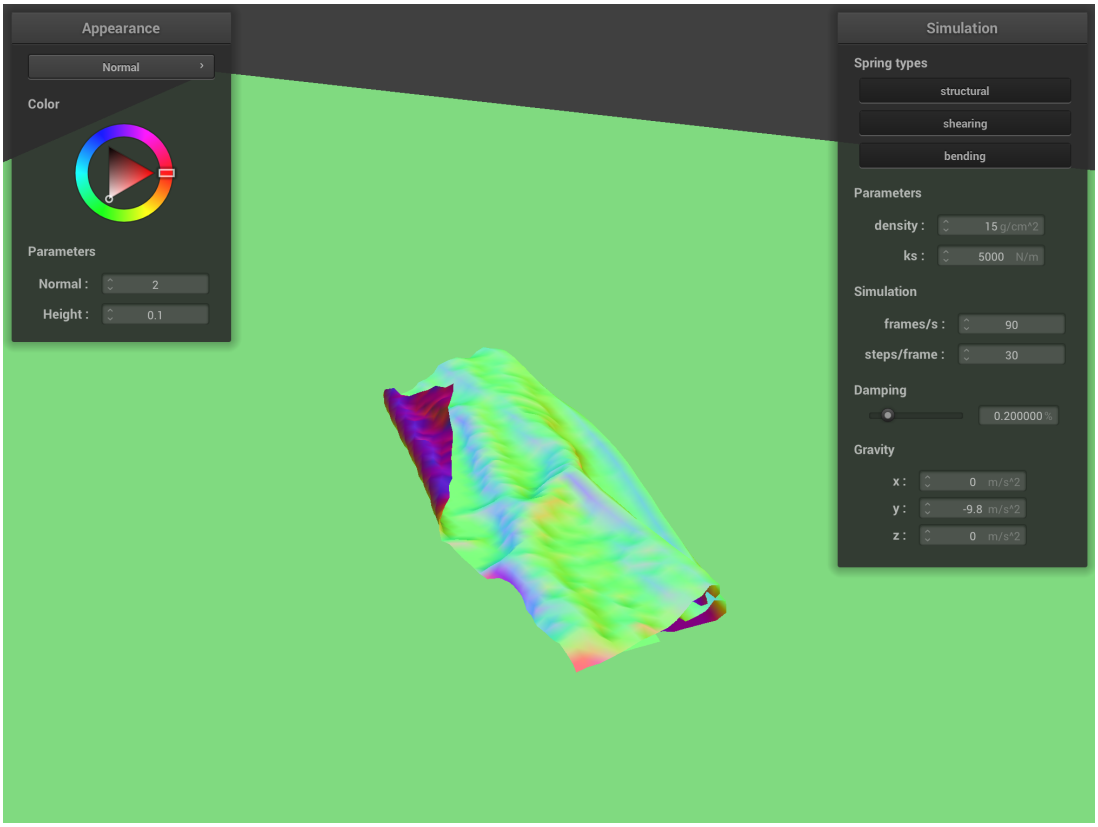
- Show us a screenshot of your shaded cloth lying peacefully at rest on the plane. If you haven't by now, feel free to express your colorful creativity with the cloth! (You will need to complete the shaders portion first to show custom colors.)

- 
Appearance
Normal
Color
Parameters
Normal : 2
Height : 0.1
Simulation
Spring types
structural
shearing
bending
Parameters
density : 15 g/cm^2
ks : 5000 N/m
Simulation
frames/s : 90
steps/frame : 30
Damping
0.200000 %
Gravity
x : 0 m/s^2
y : -9.8 m/s^2
z : 0 m/s^2
- 
Appearance
Phong
Color
Parameters
Normal : 2
Height : 0.1
Simulation
Spring types
structural
shearing
bending
Parameters
density : 15 g/cm^2
ks : 5000 N/m
Simulation
frames/s : 90
steps/frame : 30
Damping
0.200000 %
Gravity
x : 0 m/s^2
y : -9.8 m/s^2
z : 0 m/s^2

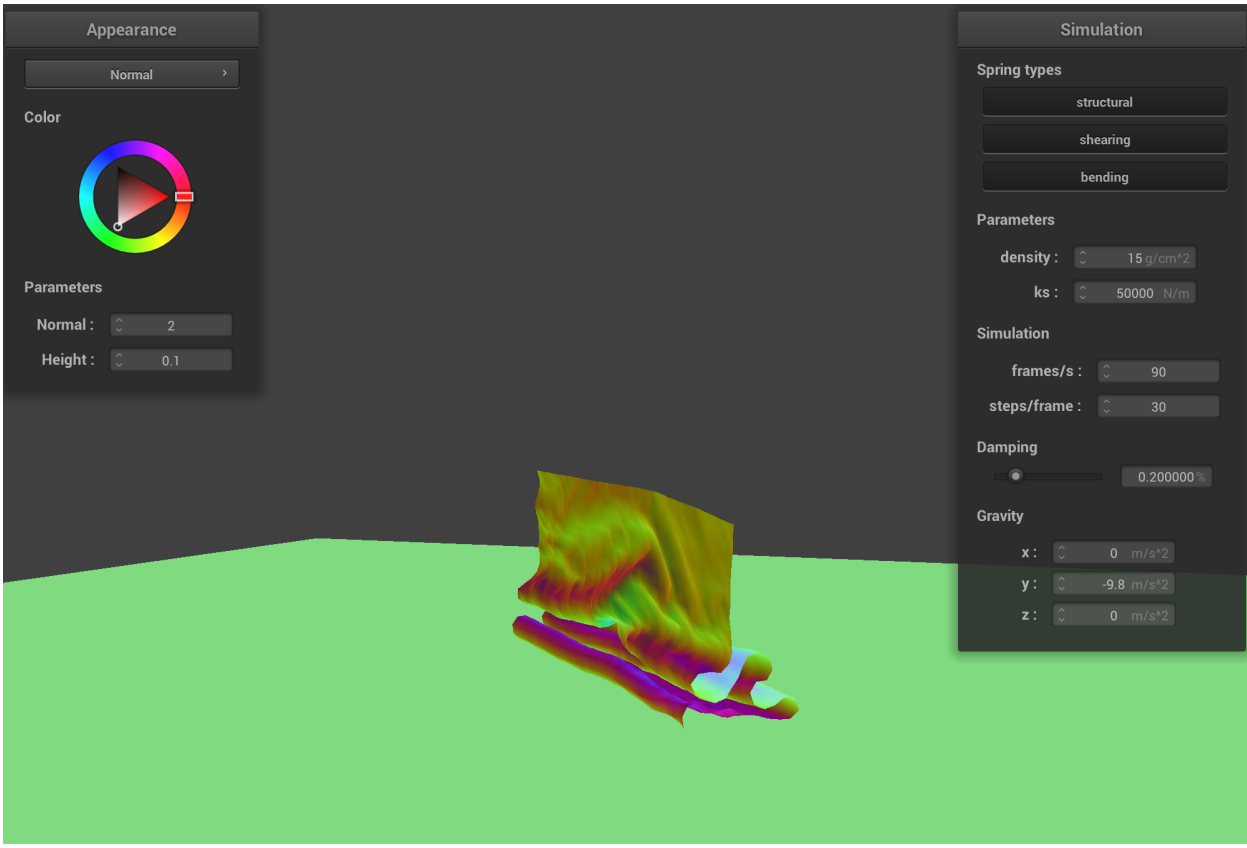
Part 4

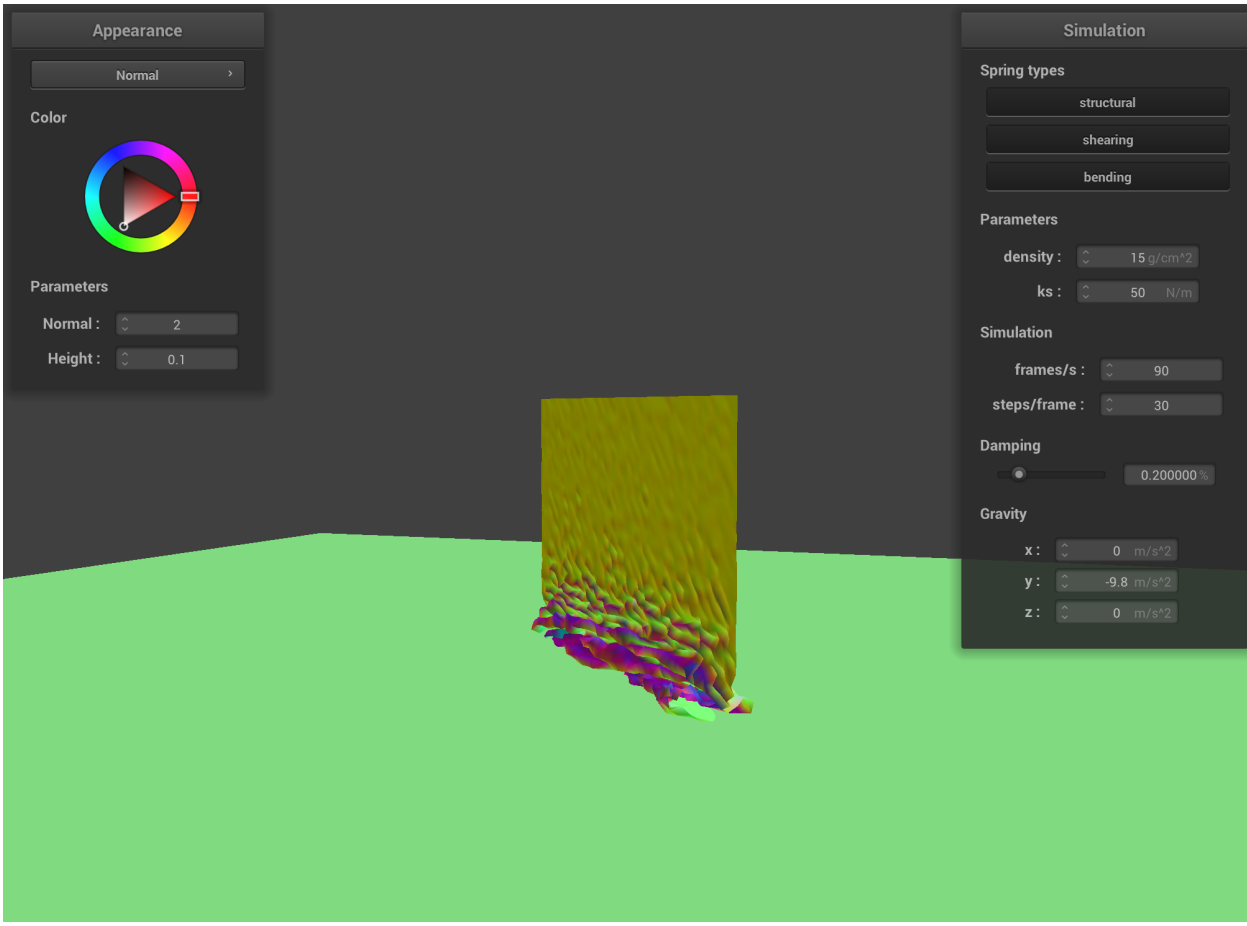
- For part 4, we used spatial hashing to hash the pointmass's position and then build a spatial map before we check for self collisions. The hash will divide it into 3D spaces into 3D boxes with dimensions of w, h, and t. After building the map, we just go through the points and check if two times the distance apart.
- 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).
  - 

- 

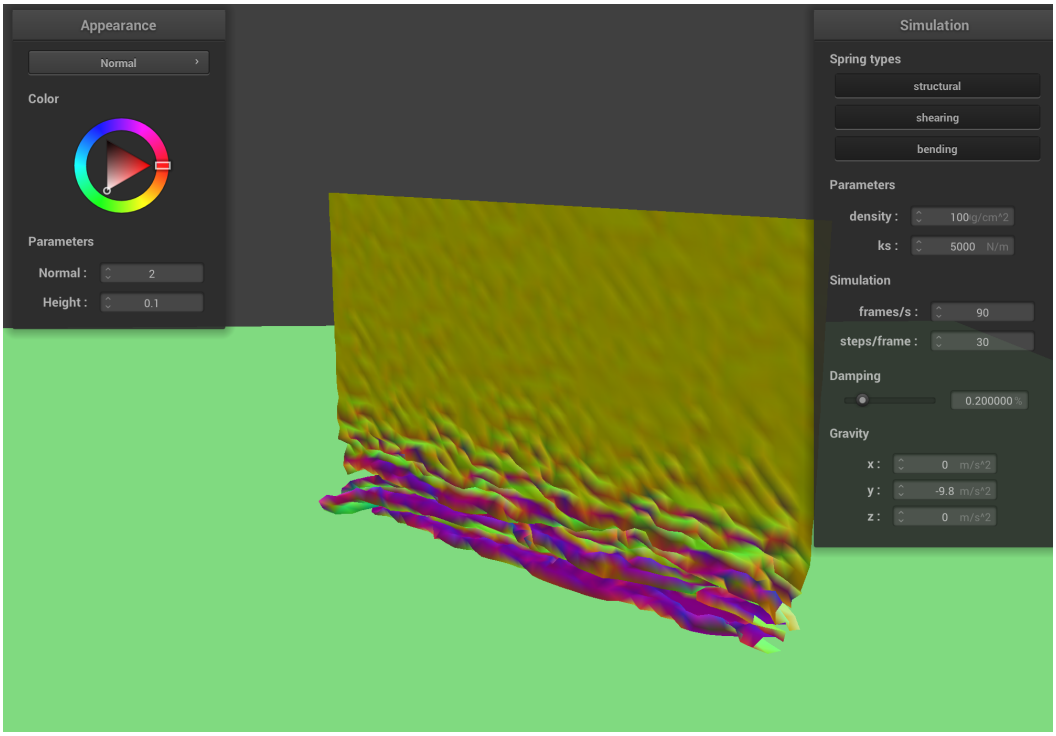
- 

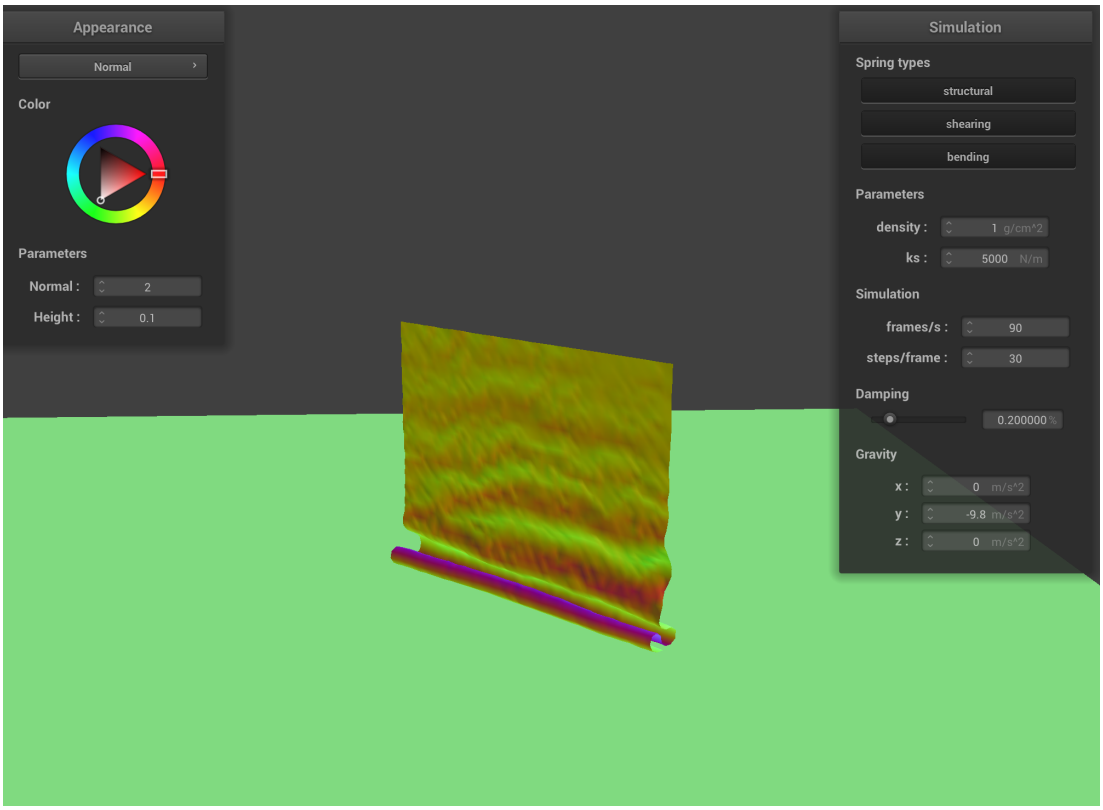
- Vary the `density` as well as `ks` and describe with words and screenshots how they affect the behavior of the cloth as it falls on itself.

- 
Appearance
Normal
Color
Parameters
Normal :
2
Height :
0.1
Simulation
Spring types
structural
shearing
bending
Parameters
density :
15 g/cm^2
ks :
50000 N/m
Simulation
frames/s :
90
steps/frame :
30
Damping
0.200000 %
Gravity
x :
0 m/s^2
y :
-9.8 m/s^2
z :
0 m/s^2

- 

- Ks:
  - High ks makes the sheet more like a thicker piece of material, more rigid
  - Low ks makes the sheet more like a thinner piece of material, more silky

- 

- 

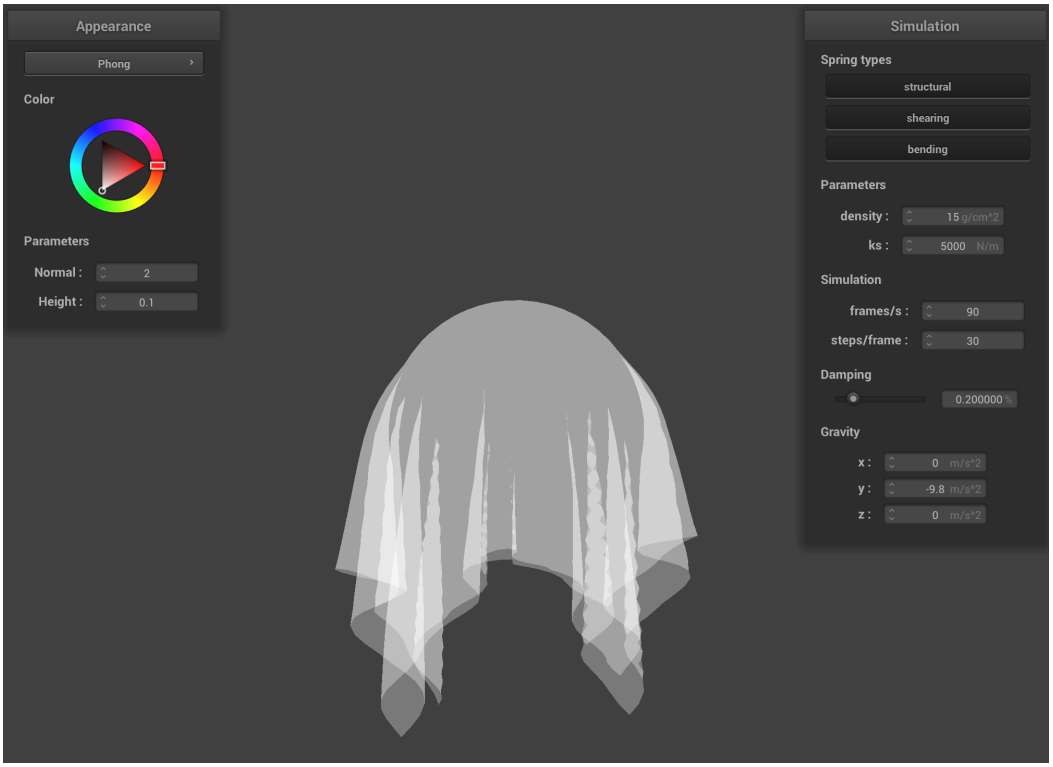
- Density:
  - High density makes the sheet seem more like silky material, falling really smoothly

        - Low density makes the sheet seem more thick, having larger curves instead of smaller curves

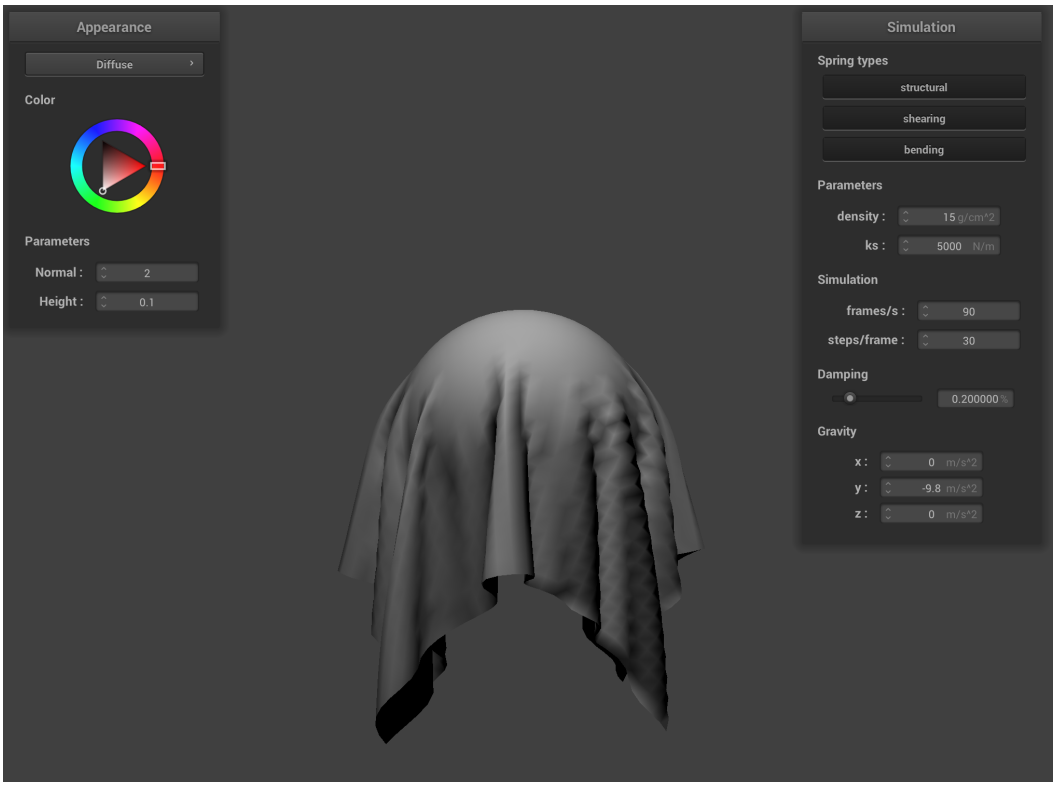
Part 5

- For part 5, we implemented shaders with GLSL, which included a lot of shaders, first we did diffuse, and then blinn phong, which is just diffuse plus ambient and specular. Texture shading would have samples where the spectrum is output as the fragment's color. The bumps and displacement shaders are very similar, where the we modify the fragment shader to shade the surface using normals, while displacement we are using the vertex shader to actually change the actual vertex, which changes the shape of it. Mirror shader we just use the eye vector to get thr reflection.
- 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 something that applies transforms to vertices, which is a vertex shader, and fragment shaders would take in geometric attributes of the fragment calculated by the vertex shader to compute and write a color. They work together by linking a vertex and fragment shader, where the output of the vertex shader becomes the input of the fragment shader.
- 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 takes three different component, ambient, diffuse, and specular, each having a specific component. Looking at the formula, you can see how each component is represented, and adding them together woud give you a great generalist shading that includes all components.

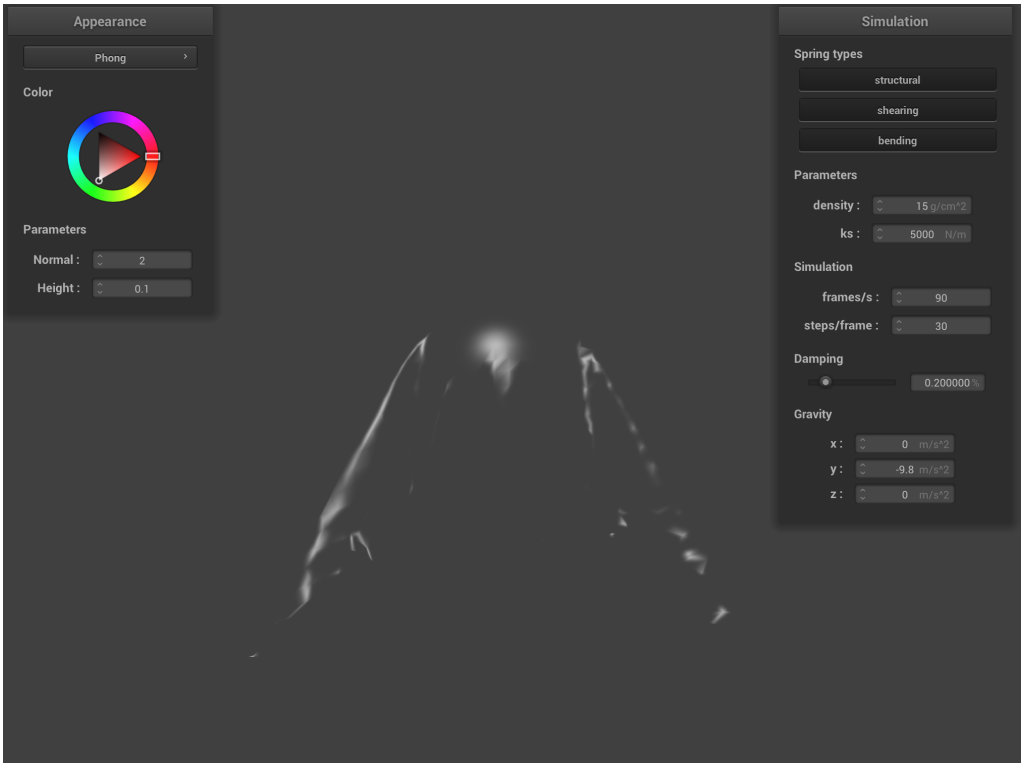
- Ambient:



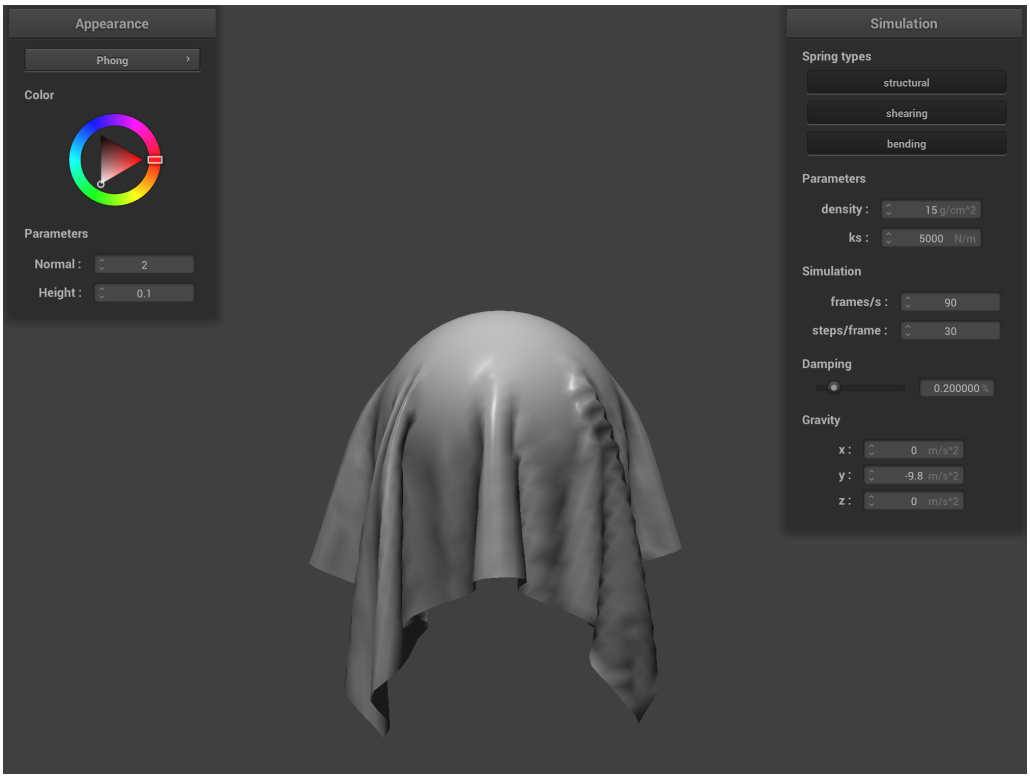
- Diffuse:



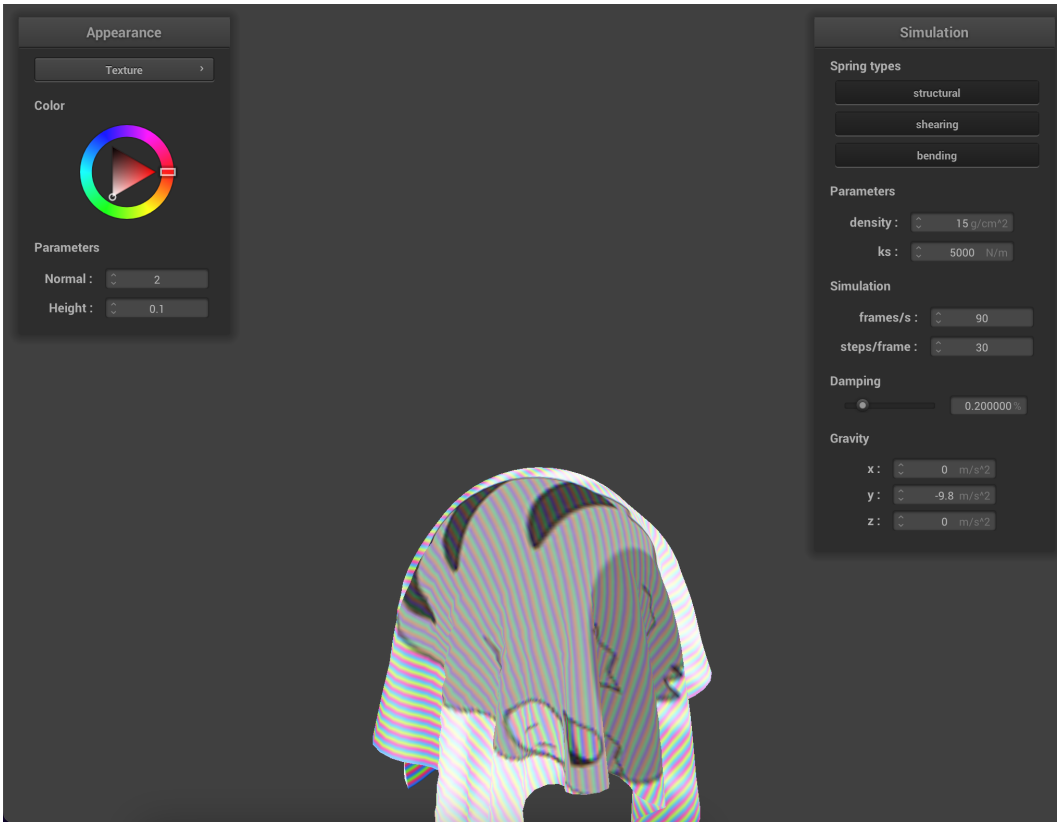
- Specular:



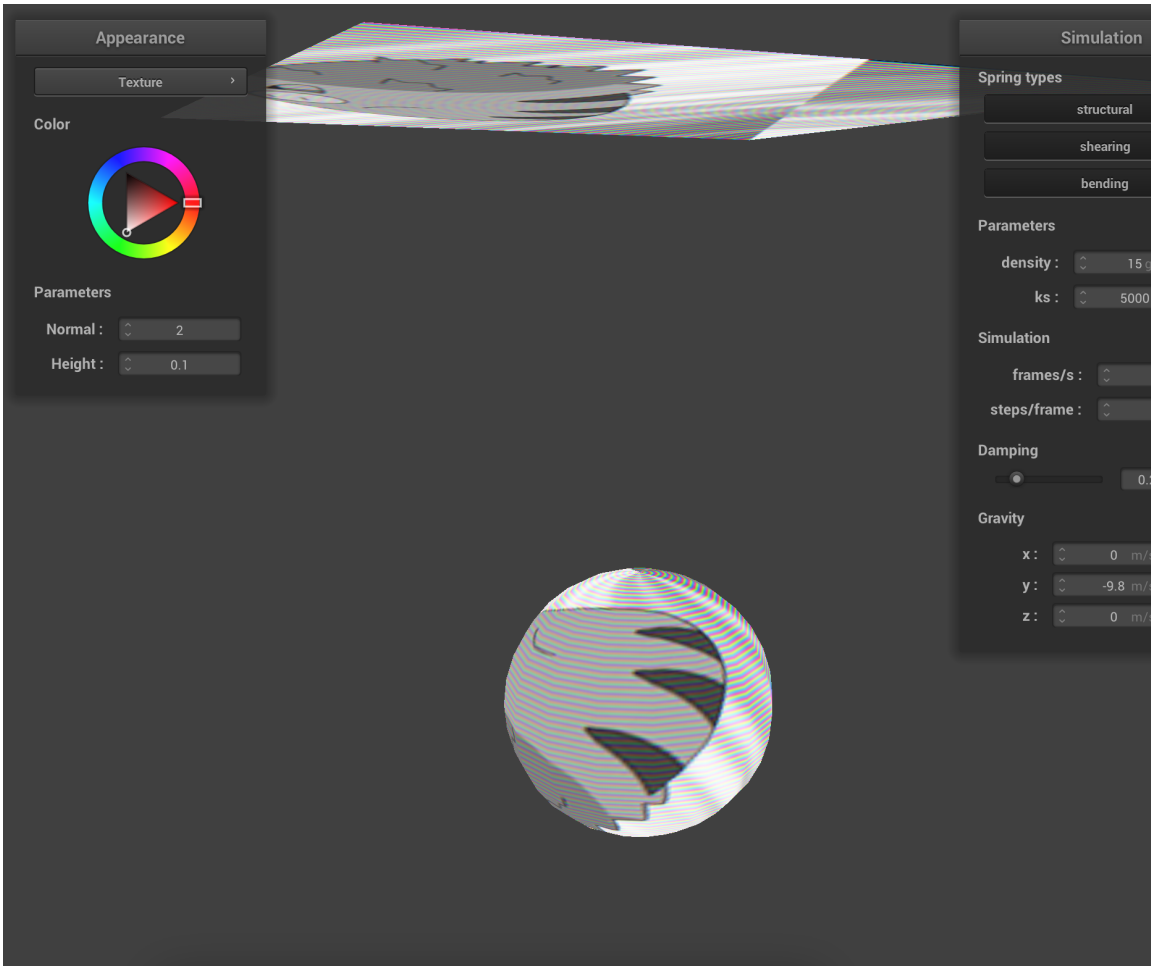
- Blinn-Phong:

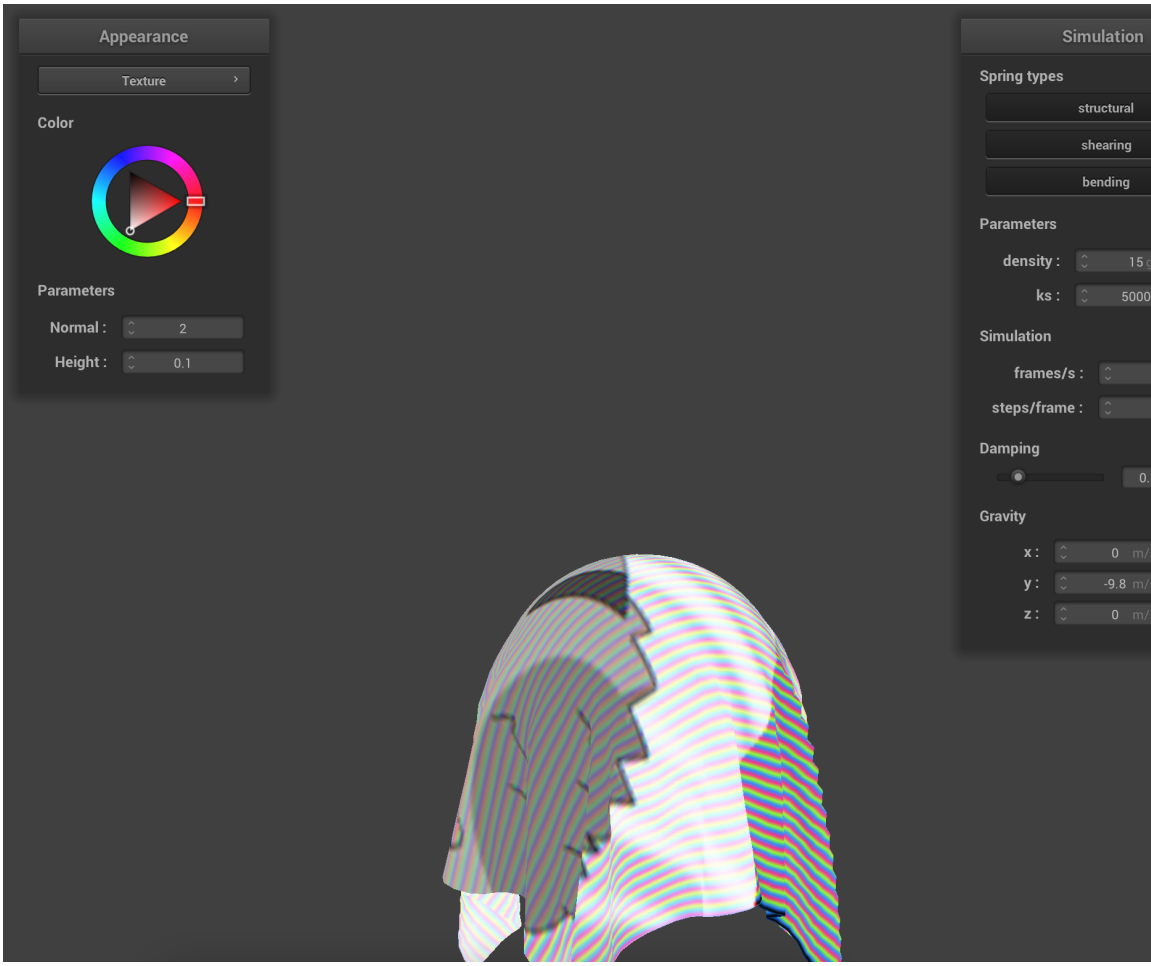


- Show a screenshot of your texture mapping shader using your own custom texture by modifying the textures in `/textures/`.

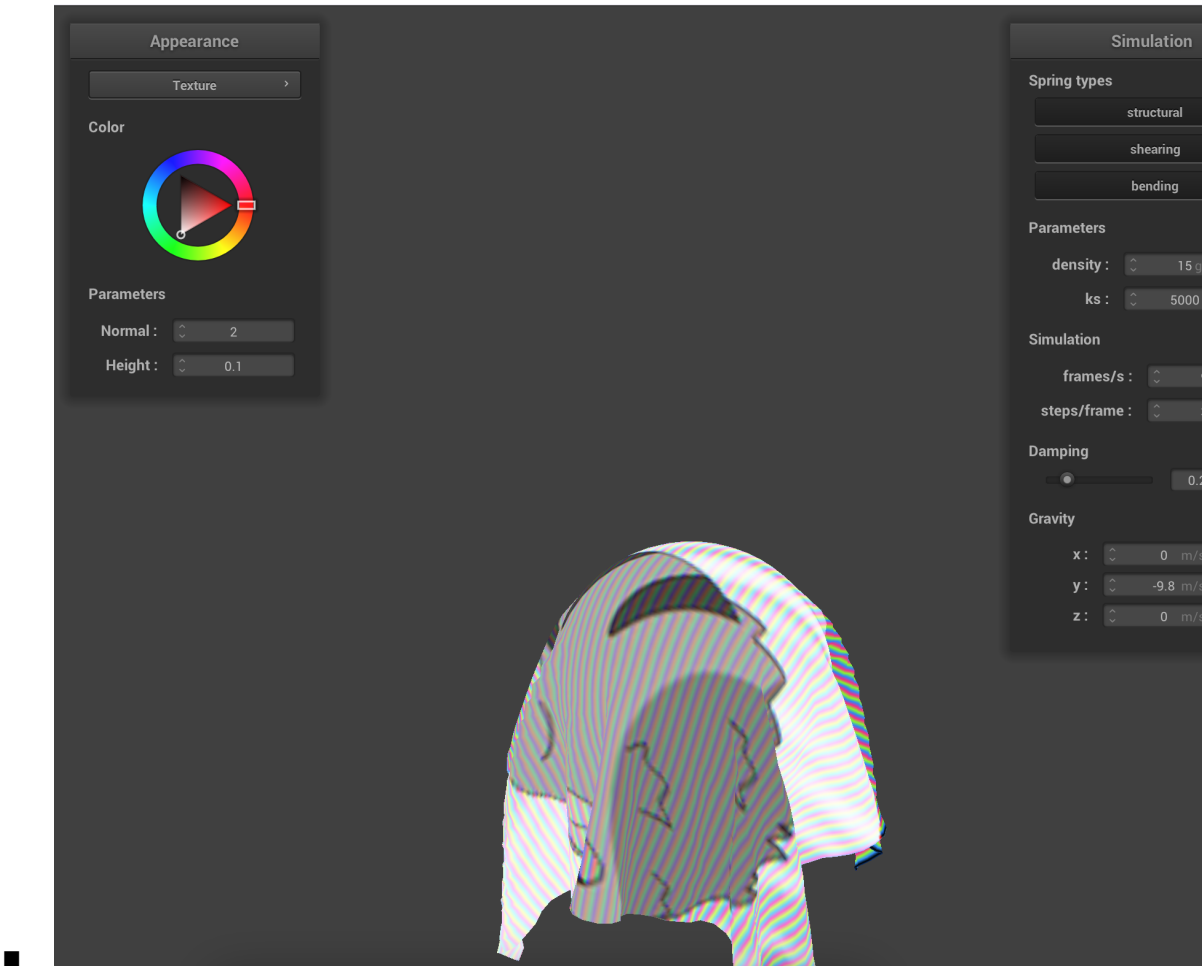
- 

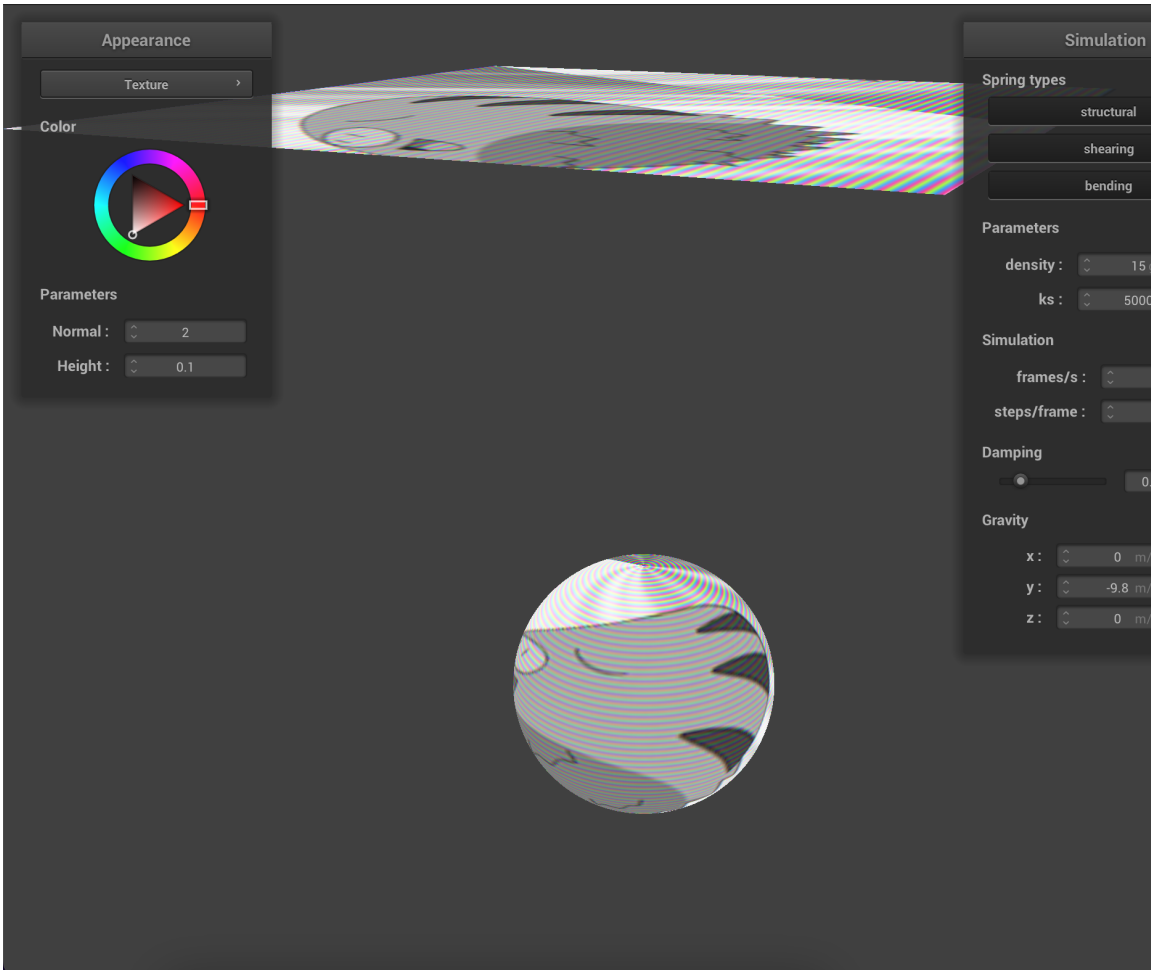
- Show a screenshot of bump mapping on the cloth and on the sphere. Show a screenshot of displacement mapping on the sphere. Use the same texture for both renders. You can either provide your own texture or use one of the ones in the textures directory, BUT choose one that's not the default `texture_2.png`. Compare the two approaches and resulting renders in your own words. 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`.
    - -o 16 -a 16


Appearance
Texture
Color
Parameters
Normal :
2
Height :
0.1
Simulation
Spring types
structural
shearing
bending
Parameters
density :
ks :
5000
Simulation
frames/s :
steps/frame :
Damping
Gravity
x :
0
y :
-9.8
z :
0

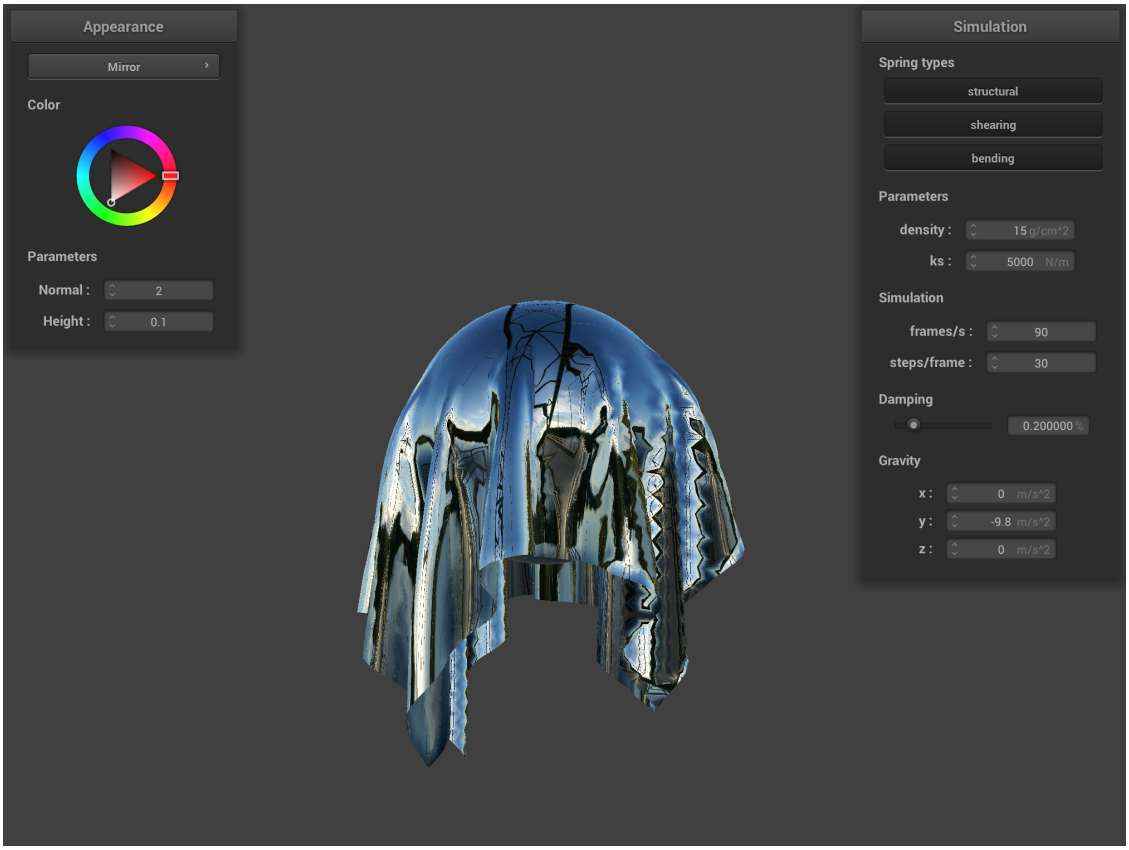

Appearance
Texture
Color
Parameters
Normal :
2
Height :
0.1
Simulation
Spring types
structural
shearing
bending
Parameters
density :
15
ks :
5000
Simulation
frames/s :
steps/frame :
Damping
Gravity
x :
0
y :
-9.8
z :
0

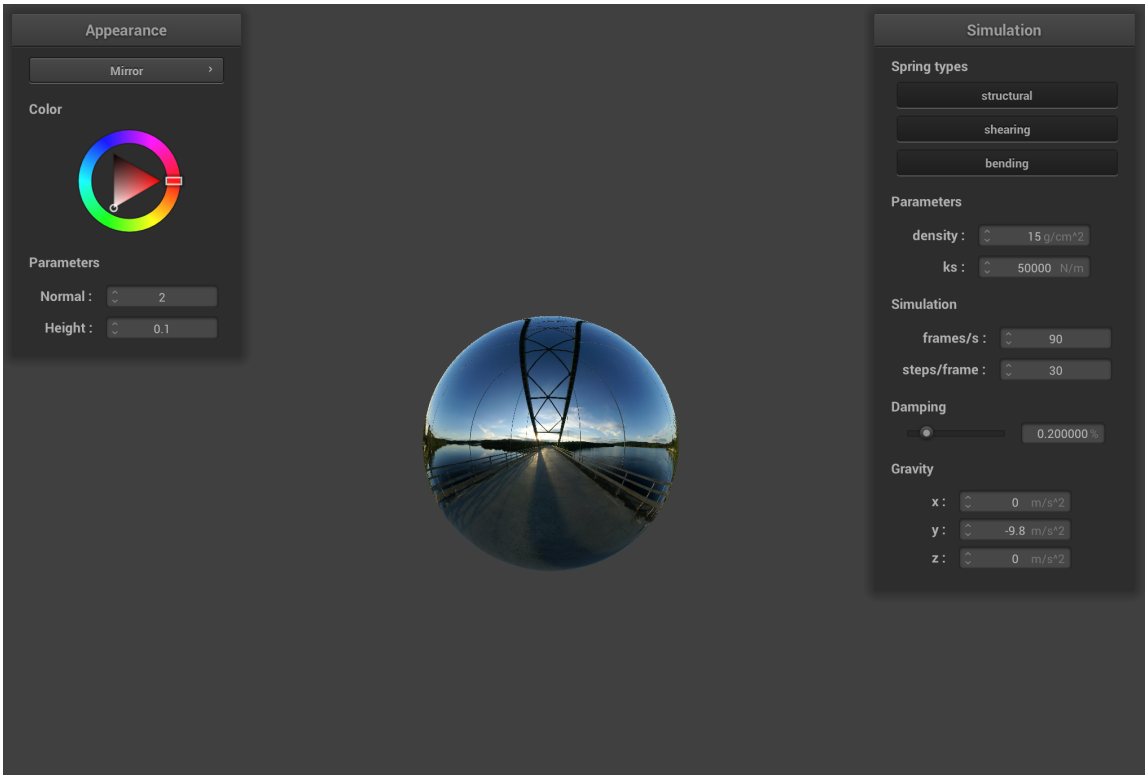
- -o 128 -a 128


Appearance
Texture
Color
Parameters
Normal :
2
Height :
0.1
Simulation
Spring types
structural
shearing
bending
Parameters
density :
15
ks :
5000
Simulation
frames/s :
steps/frame :
Damping
Gravity
x :
0
y :
-9.8
z :
0



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


Appearance
Mirror
Color
Parameters
Normal : 2
Height : 0.1
Simulation
Spring types
structural
shearing
bending
Parameters
density : 15 g/cm^2
ks : 5000 N/m
Simulation
frames/s : 90
steps/frame : 30
Damping
0.200000 %
Gravity
x : 0 m/s^2
y : -9.8 m/s^2
z : 0 m/s^2


Appearance
Mirror
Color
Parameters
Normal : 2
Height : 0.1
Simulation
Spring types
structural
shearing
bending
Parameters
density : 15 g/cm^2
ks : 50000 N/m
Simulation
frames/s : 90
steps/frame : 30
Damping
0.200000 %
Gravity
x : 0 m/s^2
y : -9.8 m/s^2
z : 0 m/s^2