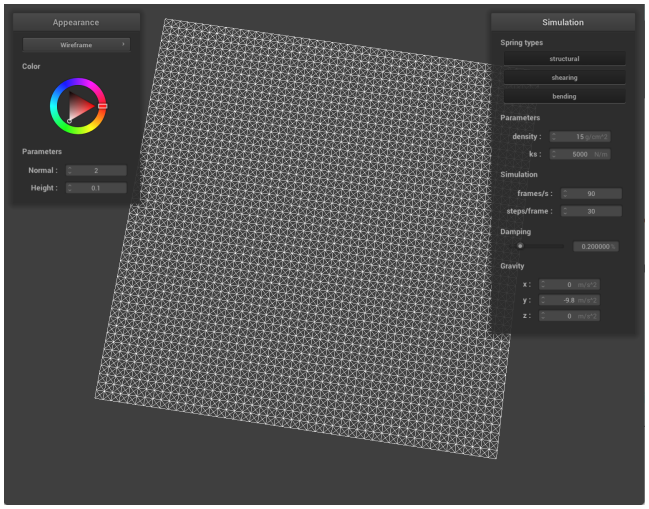
Assignment 4: Clothism

Jacob Hsiung

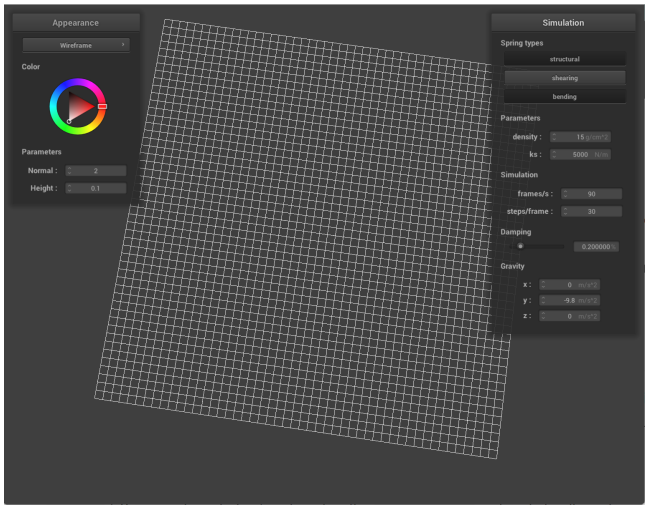
In this project, we are modeling the behavior of a piece of cloth with masses and springs. Then, we used Newton's Second Law to calculate the total external force acting on each point mass, and we also have to take into account the spring forces acting on the two ends of the spring. At the end, we use Verlet integration to compute the new point mass positions, but we also have to constrain the point massess' position such that the spring's length is at most 10% greater than its rest length. In the next part of the project, we handle the cases where the point mass is colliding with sphere and plane, so we can see the right behavior in our simulator when the cloth collide with a sphere or a plane. Aside from colllision with other object, we also need to consider the cases where the point masses are colliding with each other. We need to make sure the point masses are not stacking on one another; instead, they should be 2 * thickness apart. In the last part of the project, we wrote codes in GLSL to achieve different affect of shading, such as Blinn-Phong Shaing, texture mapping, and Displacement Mapping. After implementing the shader, we can simulate clothe with different texture.

Part 1: Masses and Springs

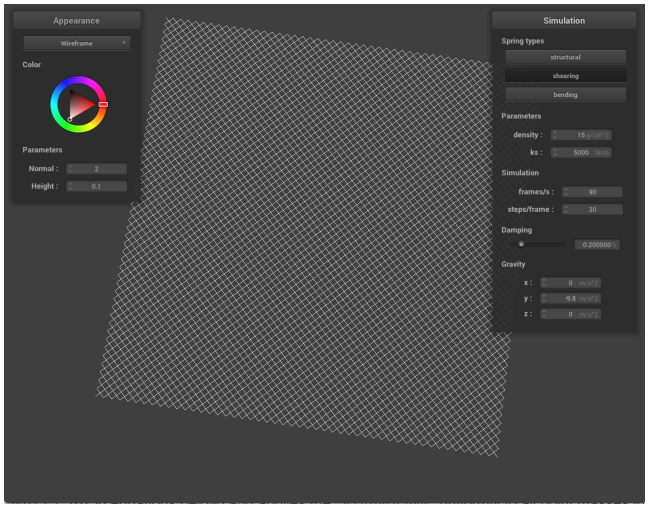
In this part, we have to create the grid according to the number of width points and the number of height points. Also, we have to apply three types of strings among the masses, including structural, shear, and bending. After populating the structure with point mass and springs, we can simulate simple effect.



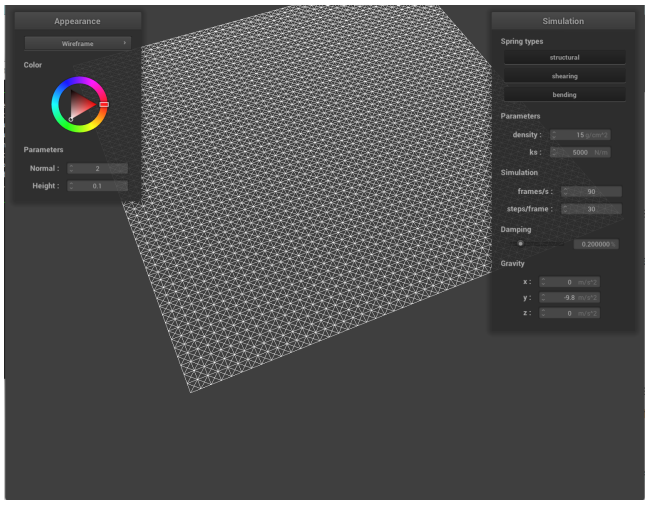
complete masses and springs



without any shearing constraints



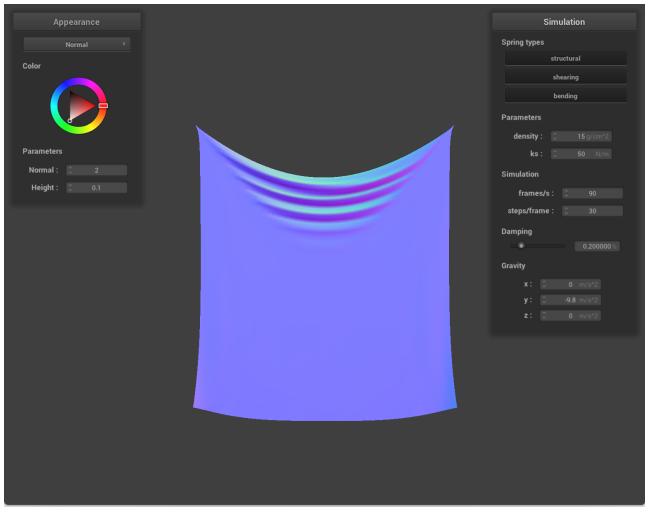
with only shearing constraints



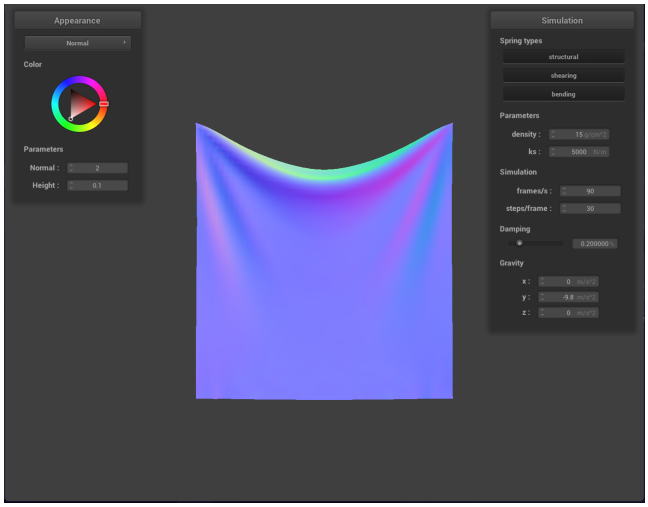
with all constraints

Part 2: Simulation via numberical integration

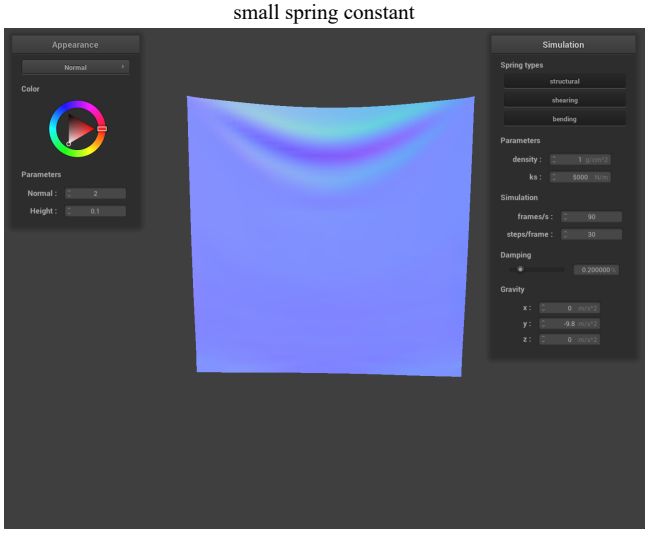
To simulate dynamic motions of the cloth, we have to calculate the change in position given the total external forces and other factors. To do so, we first calculate the total external force exerted on the cloth then calculate the acceleration onto each point mass. Furthermore, we have to also consider the force exerted by the spring due to deformation. Subsequently, we can calculate the new position using the last_position and Verlet integration. But to prevent unrealistic behavior, we have to constraint the point masses position in the end. If one of the end is pinned, then we apply changes to the other side. If both sides are pinned, then we do nothing. If both sides are free, then we apply half of the correction to each of them.



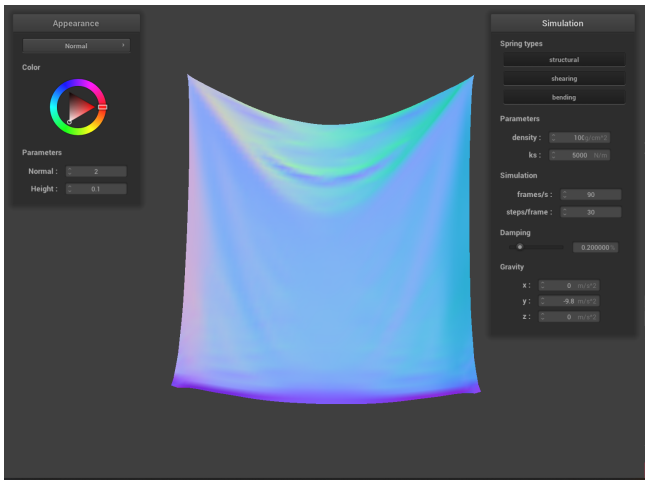
small spring constant



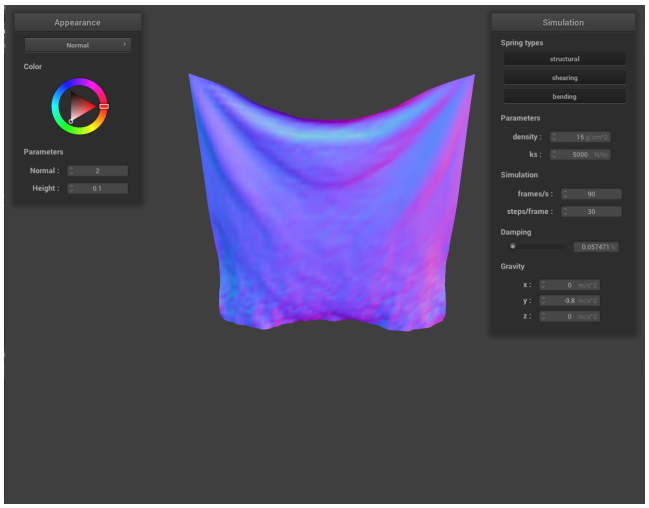
medium spring constant



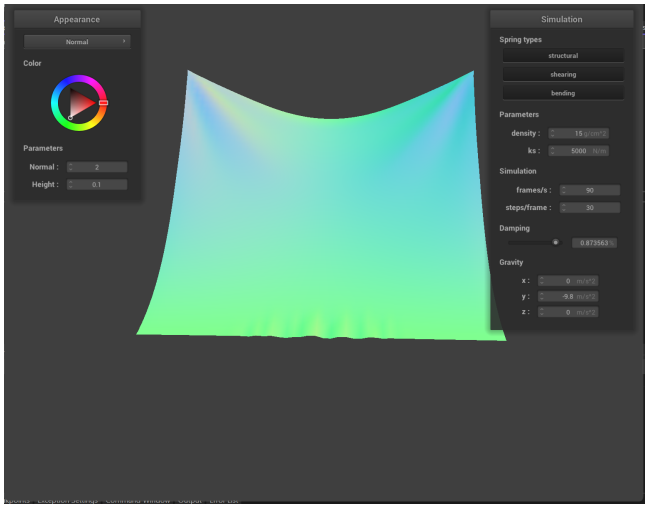
density = 1



desnity = 100



damping = 0.057



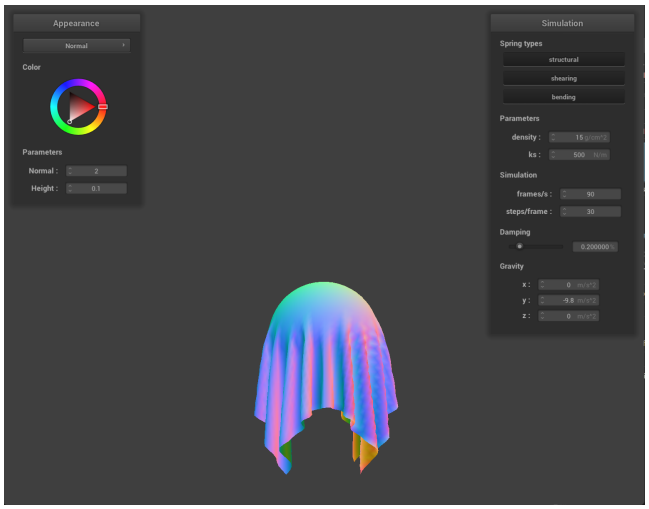
damping = 0.87

For spring constants, if a low ks is applied then the cloth will deform more at the top, when a high ks is applied, we can see that the top of the cloth barely deform. Also, when ks is low, the arc on the top of the cloth has a larger radius, and the arc is smaller for higher ks. When a small density is applied, the cloth comes to rest very fast, and the top of the cloth doesn't deform that much because of the low weight. However, when the density is large, the deformation in the top is relatively larger, and there are some swinging at the bottom of the cloth. When we apply a low damping, the cloth swings a lot when it is falling down because of low energy loss. If the damping is set to larger value, the cloth doens't really swing around when it falls.

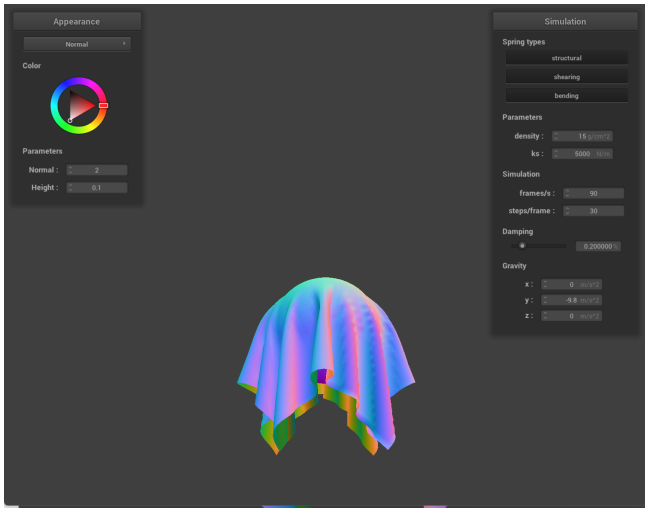
Part 3: Handling collisions with other objects

Even though now our cloth is able to exhibit dynamic behavior, but it cannot handle collision with other subjects yet. In this part, we are going to enable collision with spheres and planes. The process is somewhat similar to what we've done in pathtracer project we calculate where the point mass should have intersected the sphere or plane and update its position using the calculated correction vector scaled down by friction. However, collision with planes are little more complicated as we can end up on both sides of the plane. If the point mass's last

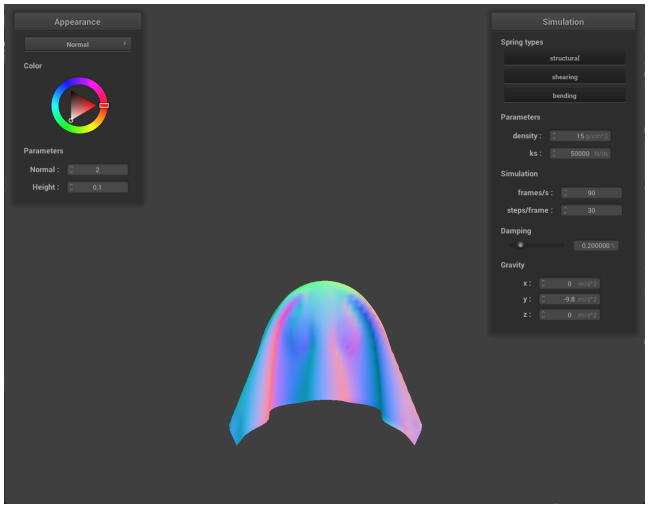
position and current position are on different side of the plane, meaning that there is a collision betweeb the point mass and the plane. Hence, we have to calculate the correction vector and apply the correction vector scaled by (1 - friction) to the last position of the point mass.



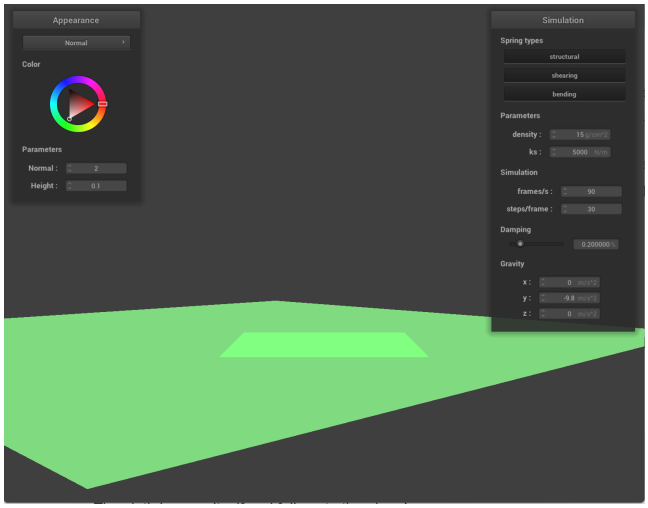
ks = 500



ks = 5000



ks = 50000

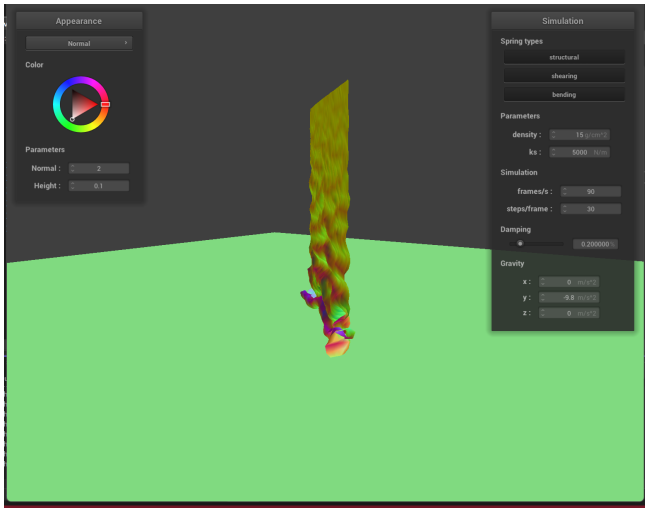


cloth collide with a plane

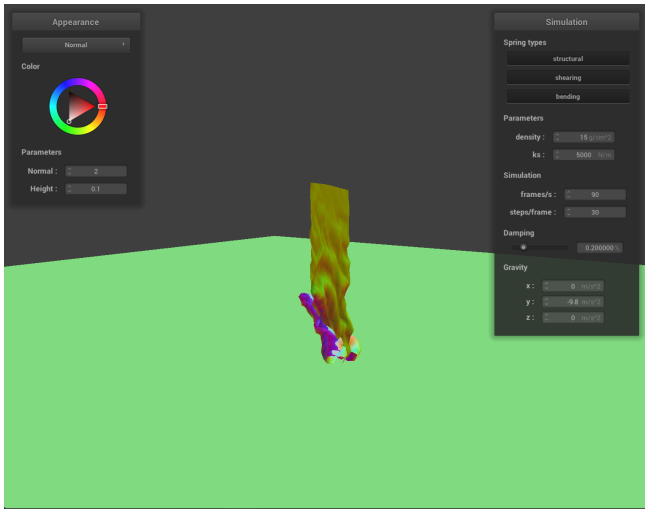
As we increase ks of the spring, it becomes harder and harder to deform the spring. So we will see less deformation of the cloth. When the spring constant is large we can see that the side of the cloth doesn't deform even if the gravity is pulling downwards.

Part 4: Handling self-collisions

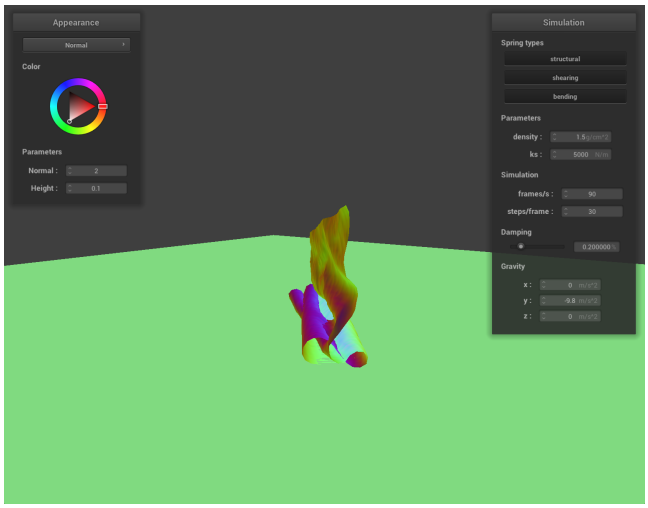
We have enable collision in the last part of our project, but our cloth now collapse on itself into a plane if we let it free fall. And this is because we didn't handle the case of self-collision. The naive method of doing so is to check every pair of point masses to ensure they are not stacking on each other which would cost lots of time. What we can do, instead, is to subdivide the space into little bounding boxes. For each point masses, we have a new set of indices corresponding to the bounding boxes they belong to. We then hash the indices and store it as a key to the an unordermap. When checking for self collisions, we only have to check the point mass with other point masses that are in the same bounding box. For every point masses, we iterate over all other point masses in the same bounding box and check if they are 2*thickness apart. If not, we calculate and accumulate the correction vector. In the end, we divide the correction vector by the number of self-collision and apply it to the point mass's position'This way we can save lots of computing power to achieve better efficiency.



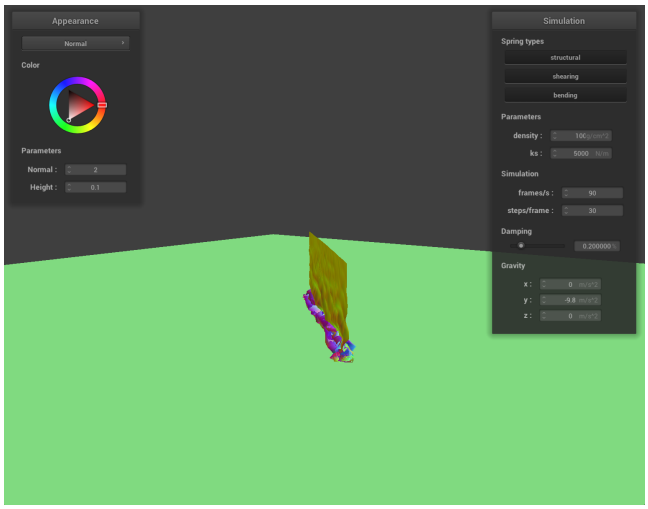
beginning of self-collision



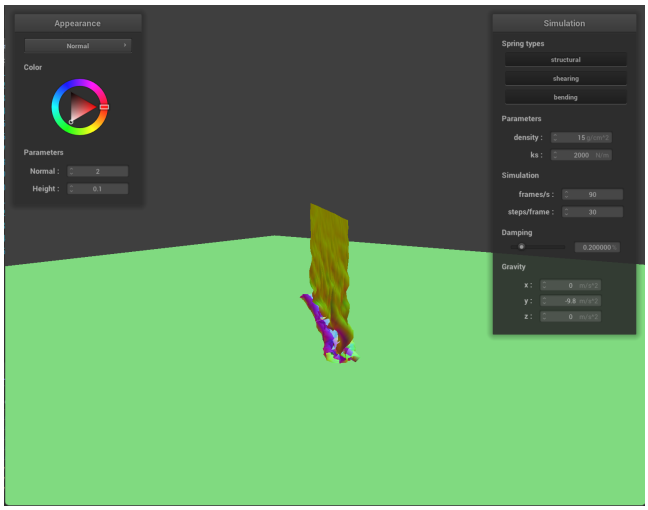
middle of self-collision



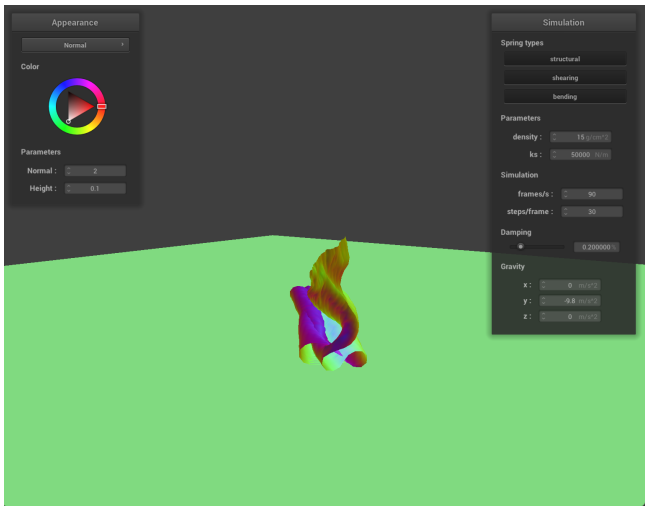
density = 1.5



desnity = 100



ks = 2000



ks = 50000

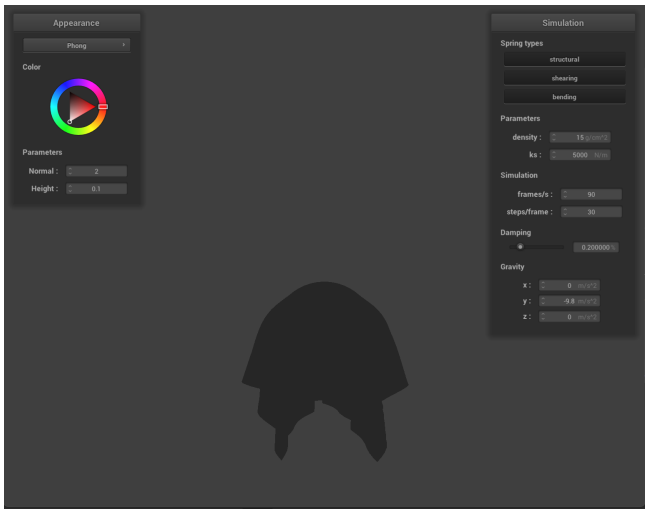
When we set density to smaller value, we see that the bottom of cloth deforms a lot more compared to when we set the density to a large value. When we set the ks to a small value, the cloth will deform with a small amount of force, so the cloth collapse on itself. However, if we set the ks to a larger value, the cloth will not deform easily, so we can see the cloth doesn't completely collapse on it self; instead, there are large wrinkles in the bottom.

Part 5: Shaders

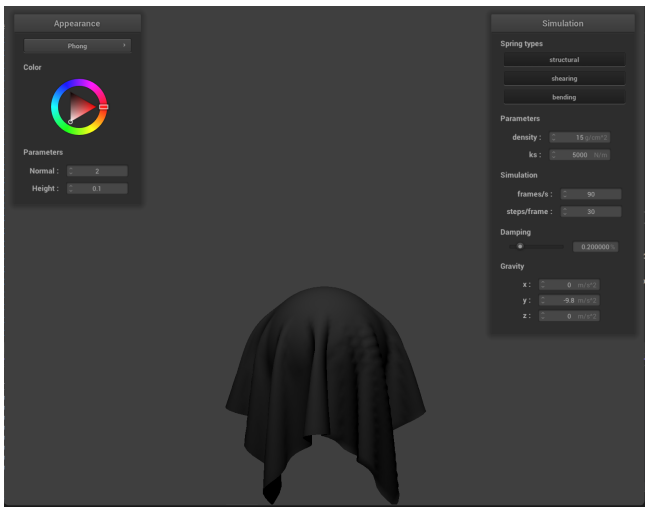
Shaders are program that computes the position, normal of a vertex position and assign it a correct color. The vertex shader is in charge of computing the position and normal of a matrix, and the fragment shader takes the output of vertex shader as a input and return the correct color.

The Blinn-Phong shading includes three term. The first time is ambient shading which is independent of the position you are in and it is uniform throughout the scene. The second term is diffuse shading, it diffuses the light and is independent of the viewing direction. The last

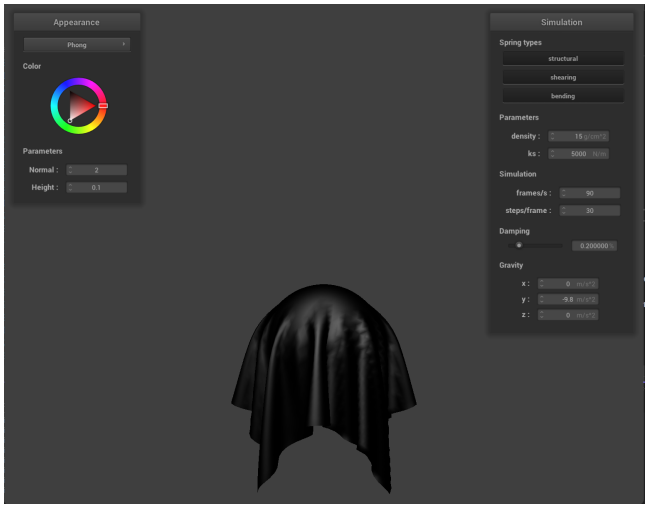
term is specular shading. It reflects light based on how close the eye direction is to the reflect direction. We return the total of the three shading added together.



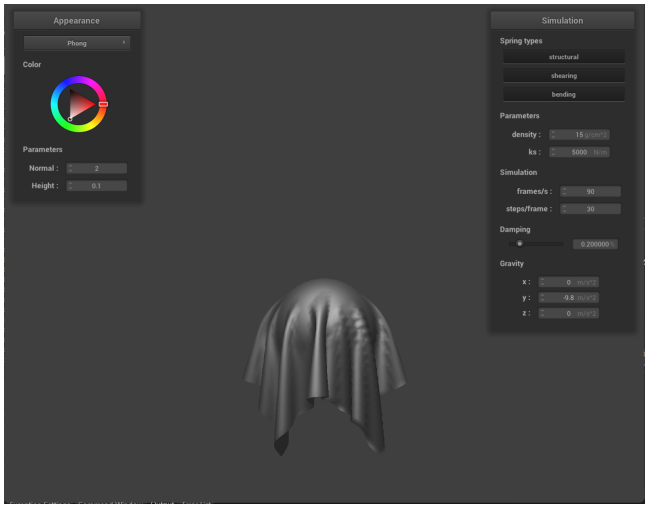
ambient shading only



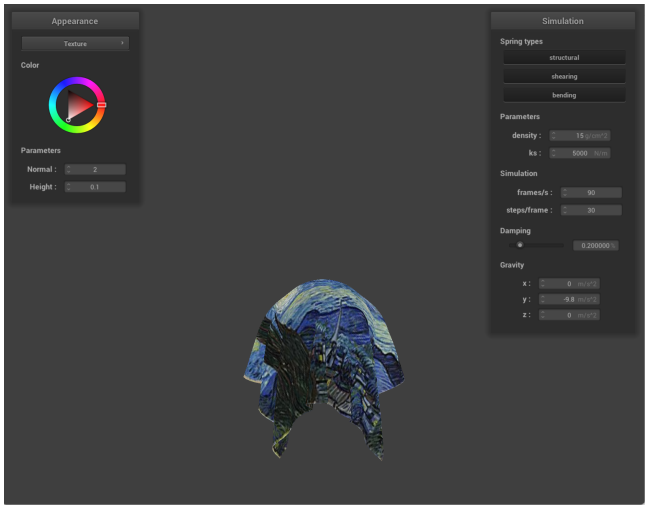
diffuse shading only



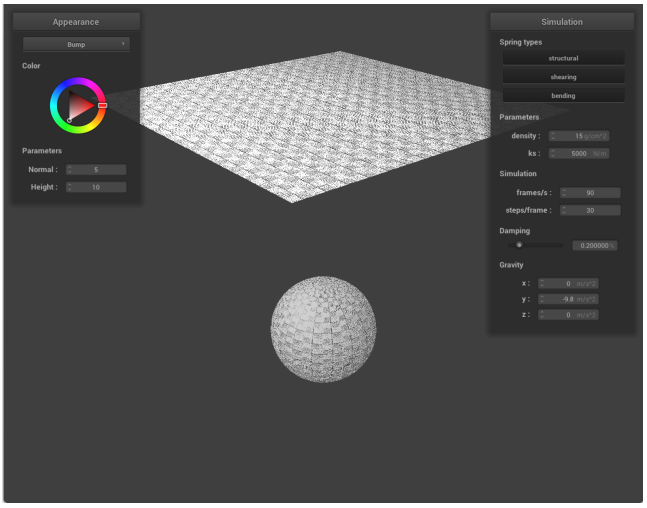
specular shading only



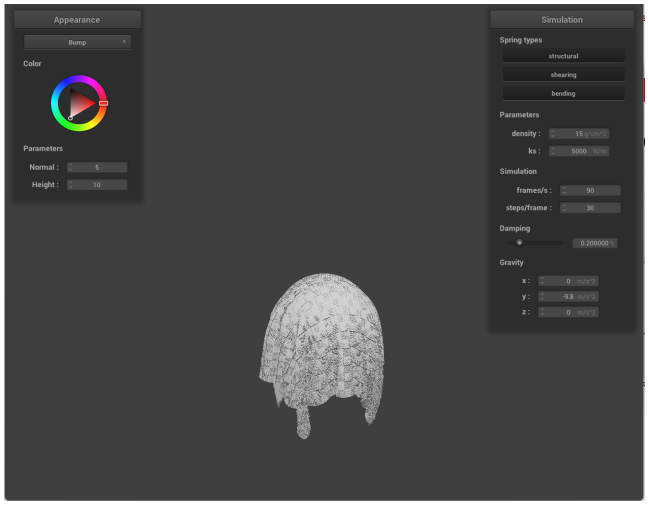
Blinn-Phone shading



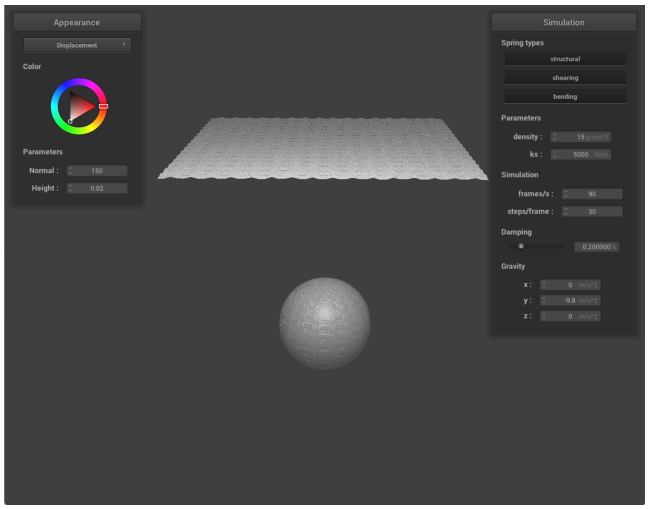
Using The Starry Night as texture



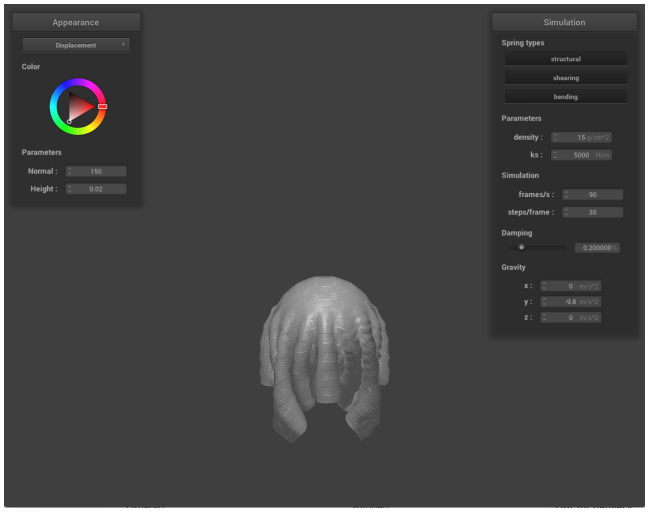
cloth and sphere of bump mapping



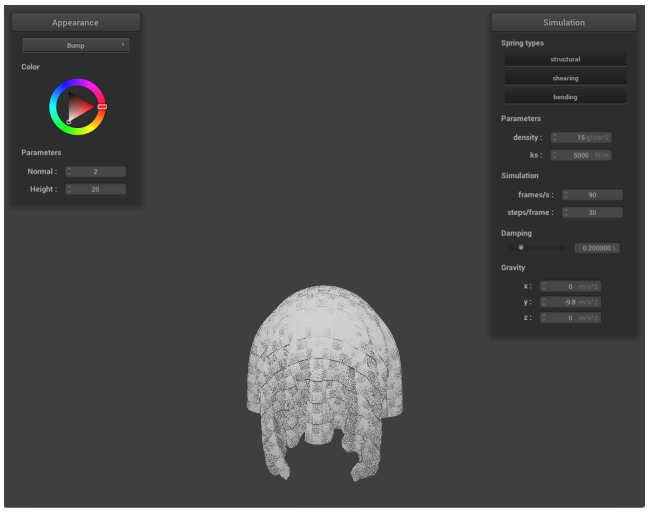
cloth on the sphere in bump mapping



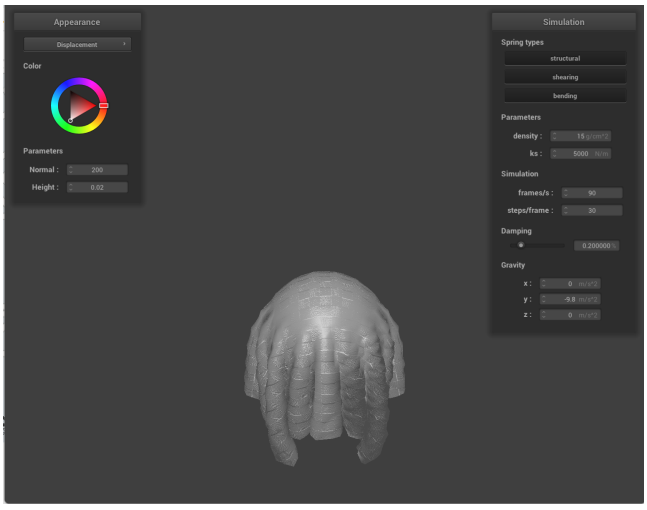
cloth and sphere in displacement mapping



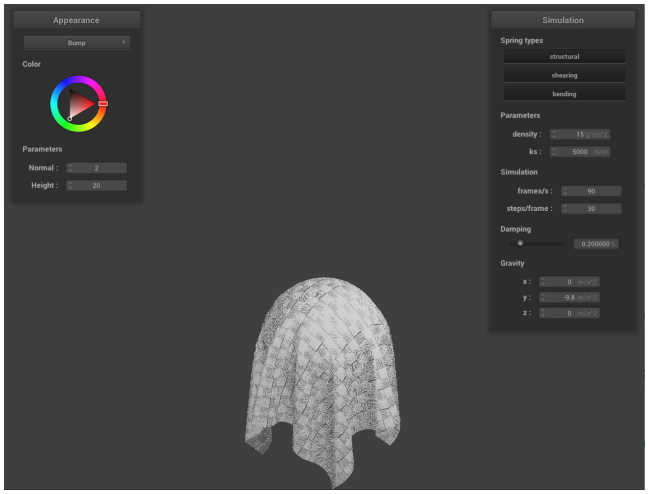
cloth on the sphere in displacement mapping



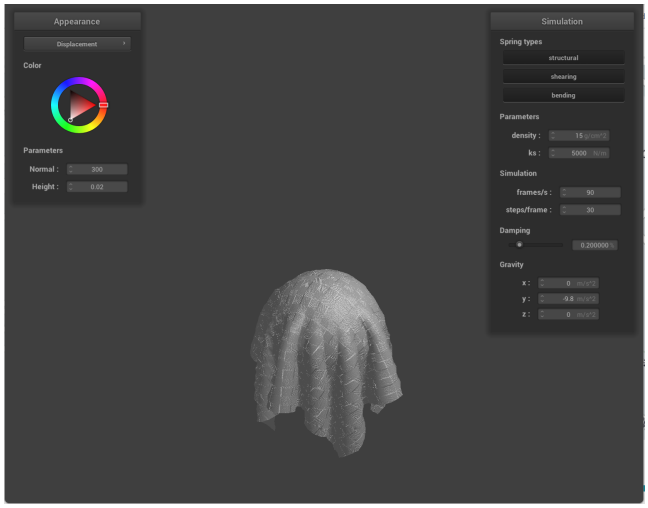
with -o 16 -a 16



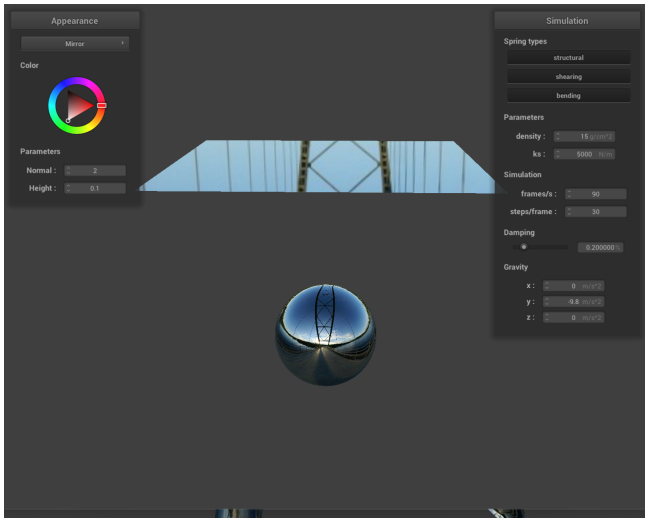
with -o 16 -a 16



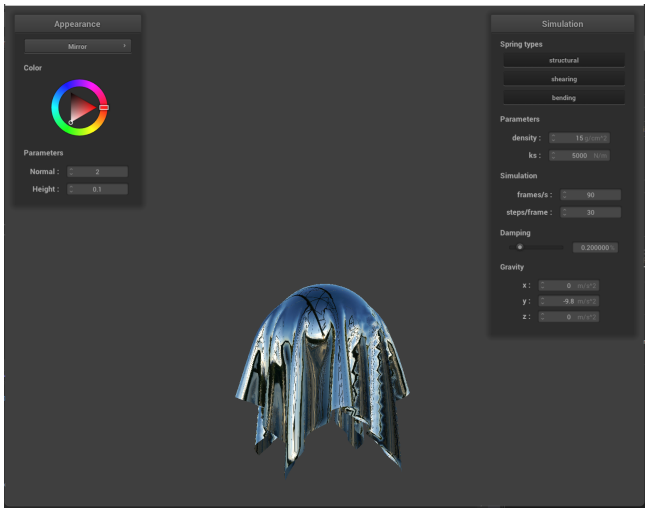
with -o 128 -a 128



with -o 128 -a 128



beginning



end

Bump mapping only changes the normal according to the height of the texture, but Displacement mapping not only changes the normal but also the vertex position. This way the texture and the position will be more consistent with each other. With -o 16 -a 16, the mapping is much more coarser than -o 128 -a 128. Also there is a lot of aliasing effect in the former image. Also, the shape of ther sphere is smoother when we set it to 128.