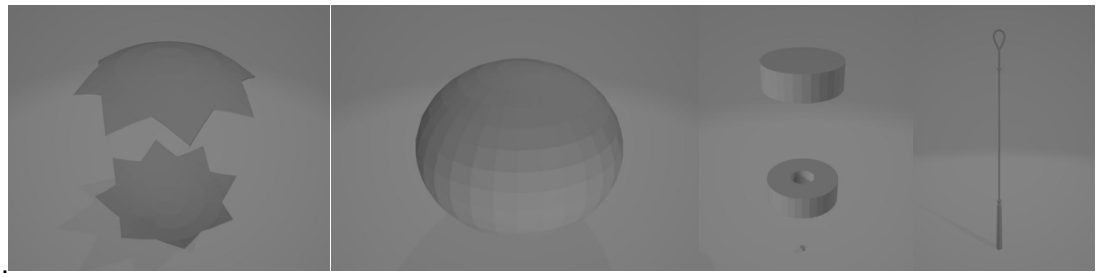
2.Rendering

First, I exported the model built in OpenJscad, but when using threejs, I found that the stl file exported from OpenJscad only contains vertex and surface data. It is very difficult to assign materials and textures to such an object, so I imported the parts with different materials into threejs for processing. It can be roughly divided into the surface texture of lanterns, the texture of cloth, the texture of metal and the texture of lines. While exporting, I forgot a small decoration on top of the lantern, which will be added in another method later. As shown in the figure below, the entire lantern is split into widgets and imported into threejs for processing.

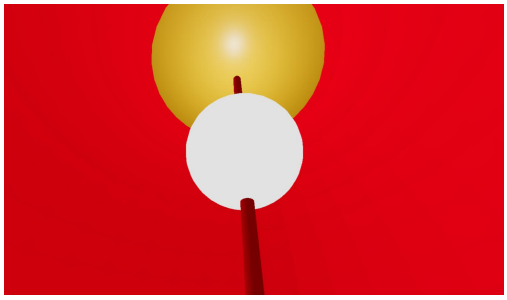


To maintain the unity of the overall rendering style, we choose to use THREE.MeshPhysicalMaterial for the material of each part. After some more debugging, we found the right material for the lantern. In the process of debugging, we found that due to the lack of UV information of the STL model, there will be a lot of errors when adding textures to the model. So, we chose to test the texture with a small decoration on top of the lantern.

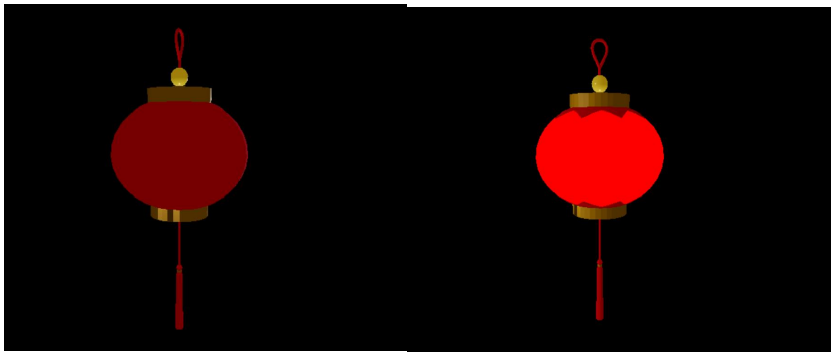
The figure below is the final surface effect. We used the built-in SphereGeometry of

threejs, and also selected MeshPhysicalMaterial to test the displacementMap, roughnessMap, metalnessMap and normalMap, and got the effect as shown in the figure.

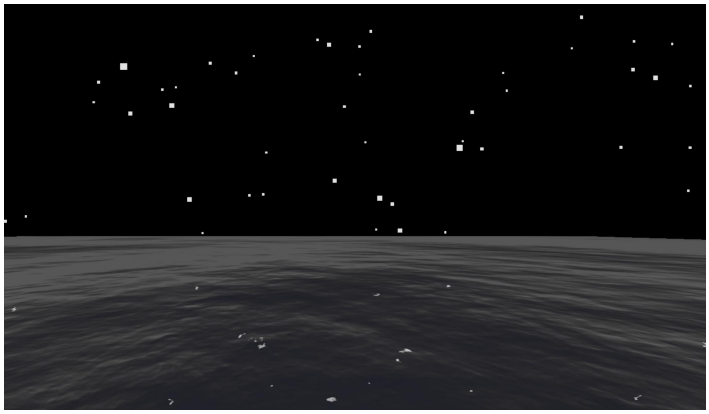
Since it is a lantern, it should have the function of emitting light. We added a point light inside the lantern.



After adding a point light source, the lantern still does not glow. This is because the light transmittance of the lantern material is not set. After setting the light transmittance to 4, the lantern successfully achieves the glow effect.



However, the background is still too empty, so we used SpriteMaterial and threejs built-in water to make a simple background.



The figure below is the presentation of the overall effect after the final rendering.