Group 4

Final Project Report

Menbers：

1、Modeling：Vicky-Wanqi Zhang

2、Animation：Lily-Yiwen Hu

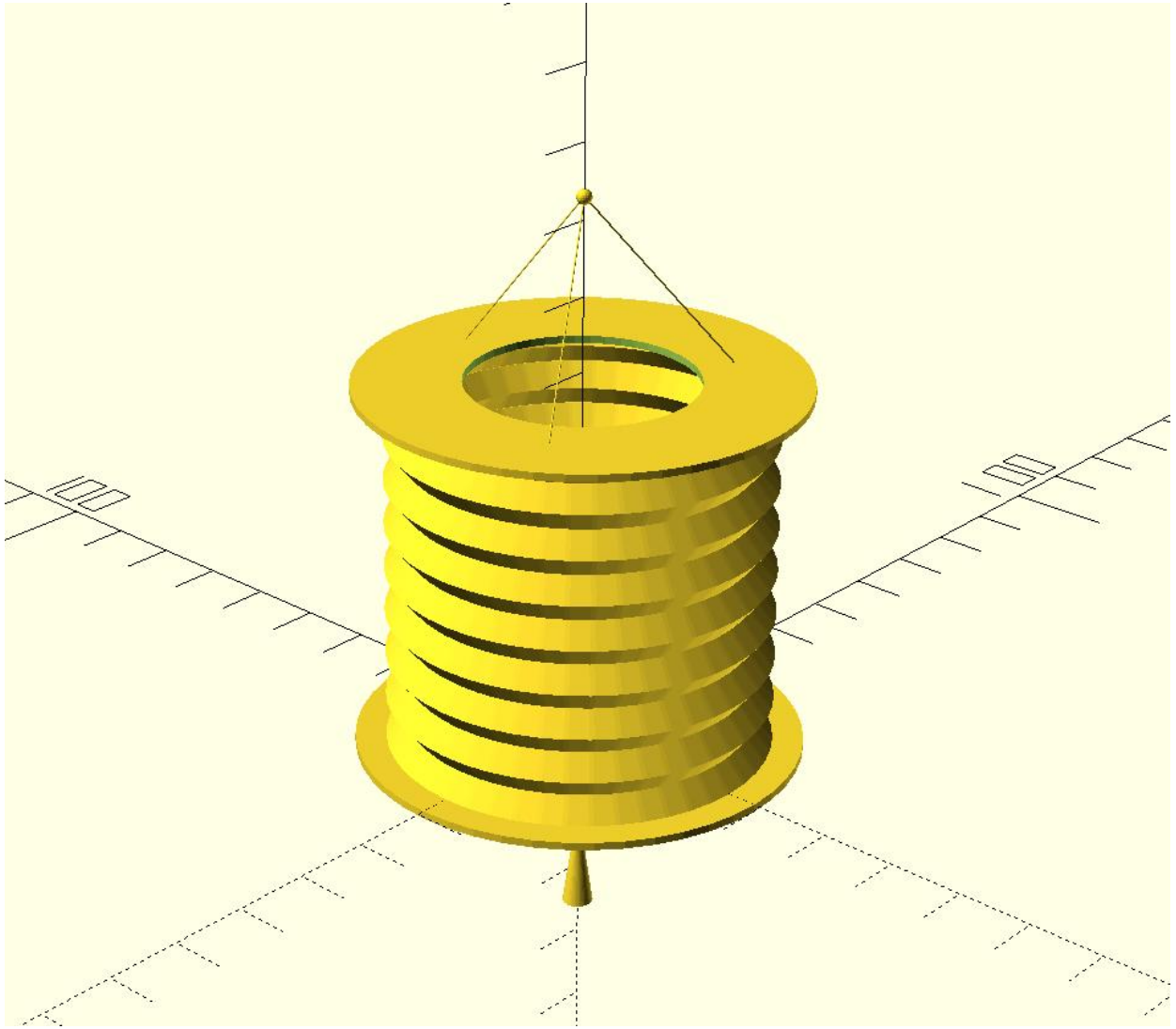
3、Materials：Cloney-Jing Nie

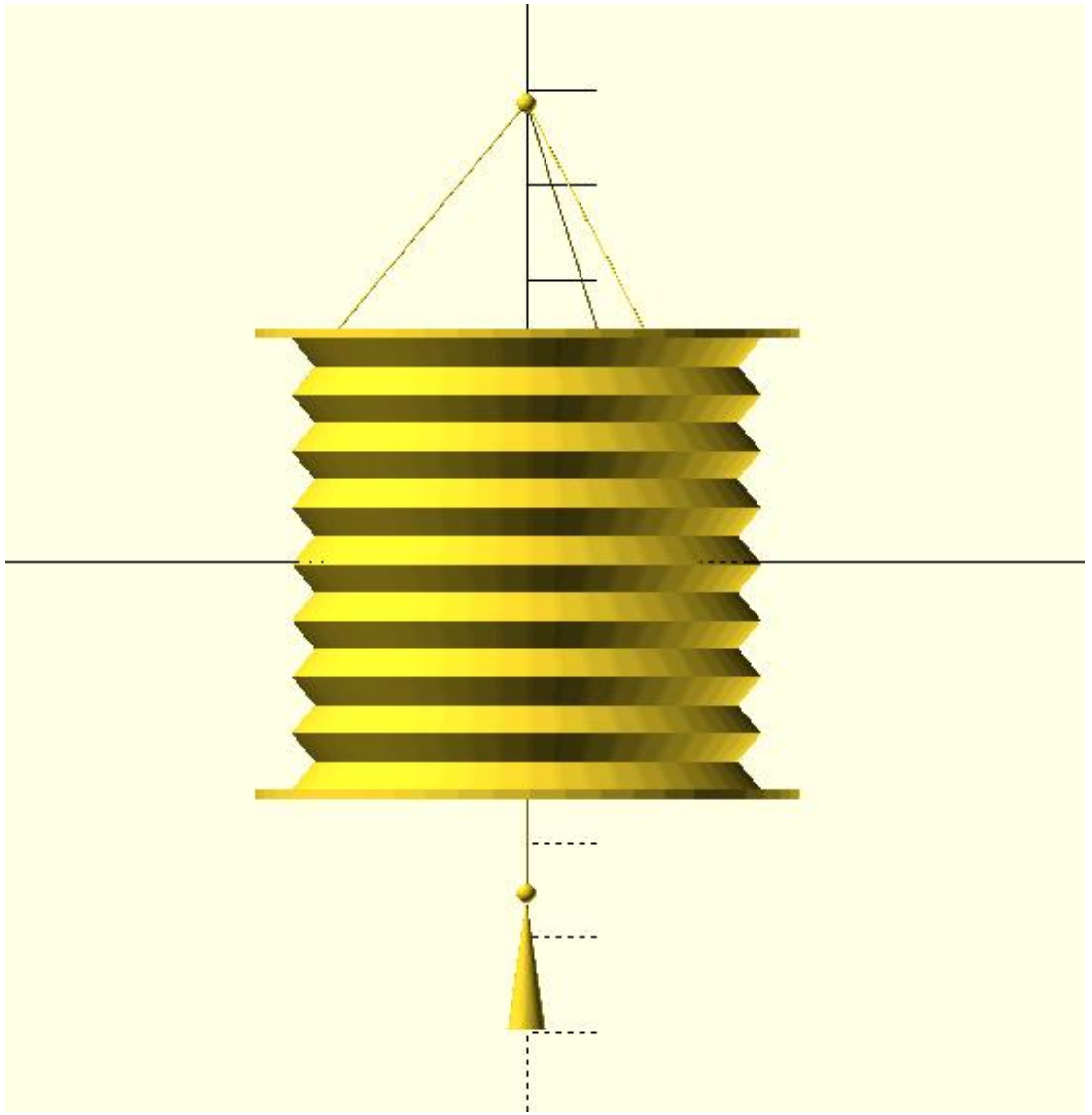
4、Camera：kelvin-HuiLin Guan

MODELING PART

Result:

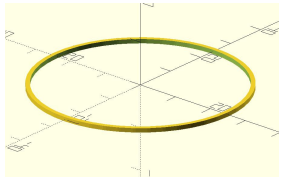
Middle part + upper part + lower part + adjusting the position



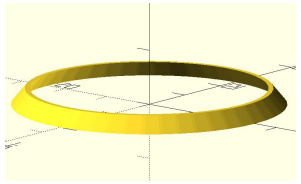


1. Middle part

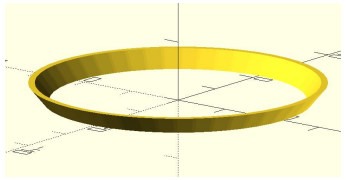
First, make a circle, then using difference() to get a ring



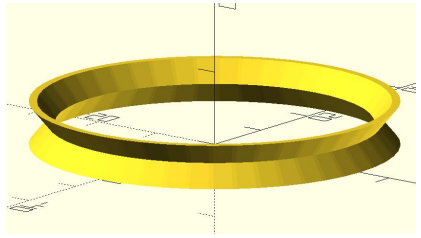
Then use linear-extrude, making it like this:



Then make the upper part.

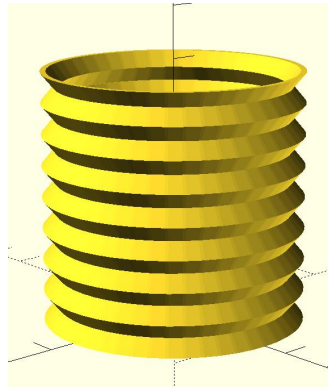


Then translate it. Put this two model together.
Making this a unit model called “basic”.

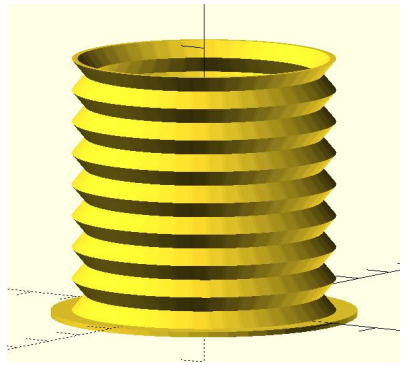


Using a loop to put many “basic” s together.

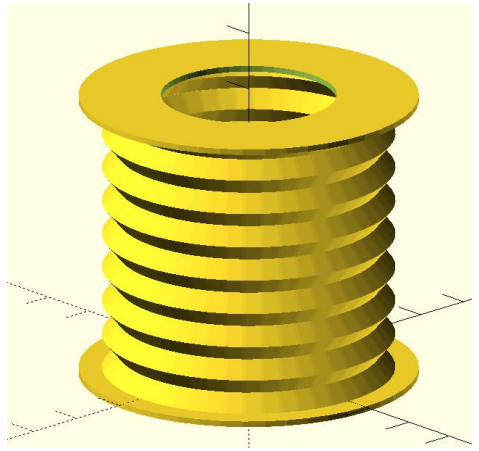
```
45 for(count=[0:1:repeattime-1]){
46 translate([0,0,count*calhei-tr])
47 basic();
48
49     }
```



Then we have to generate a plate under these “basic” s.
Generate a circle, then use linear-extrude() to make it a Cylinder, which looks like a plate.
Then translate this plate down “basic” s.



Then generate the upper plate:
Generate a circle, using linear-extrude() to make it a Cylinder. Use the same way making a smaller cylinder.
Then use difference() to make a ring. Translate this ring Higher than those “basic” s.

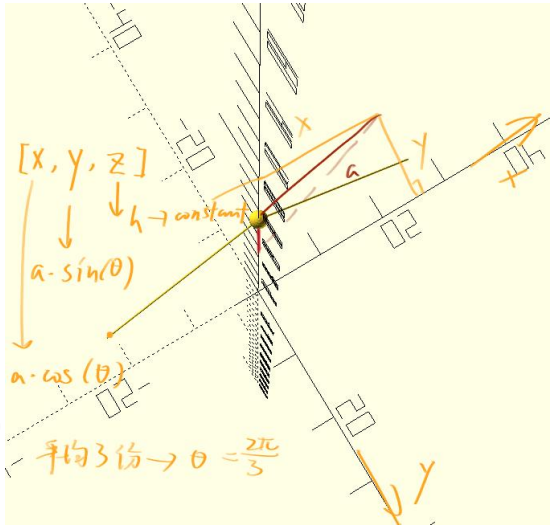


2. Upper part

2 parts: 3 lines+1 sphere

3 lines:

First, generate 3 cylinder/sphere at the position on the top of middle part.We know z is 48. we have to determine x and y.



For the first cylinder:

X will be radius * cos(360/3)

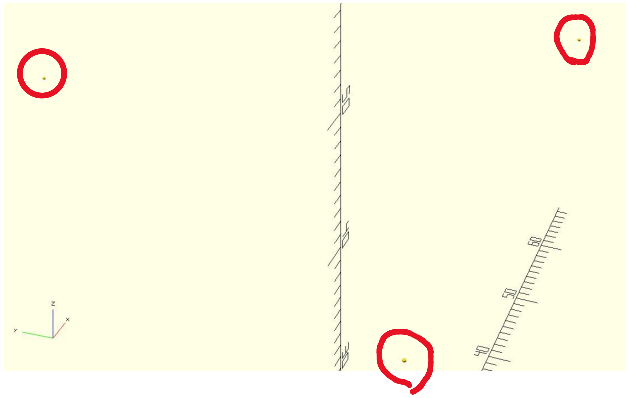
Y will be radius * sin(360/3)

For the Second cylinder:

X will be radius * cos(2*360/3)

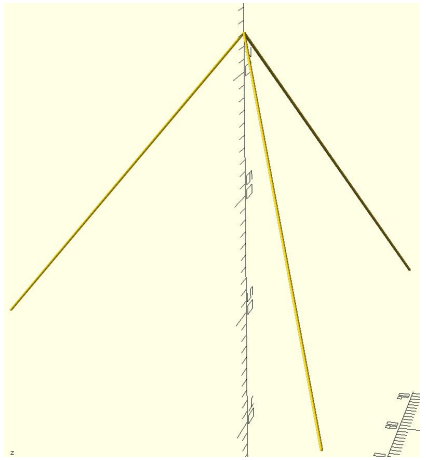
Y will be radius * sin(2*360/3)

Then, we get three cylinder at apt position.

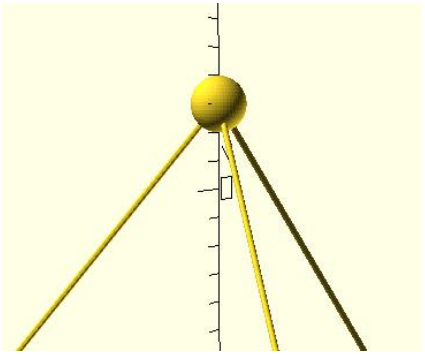


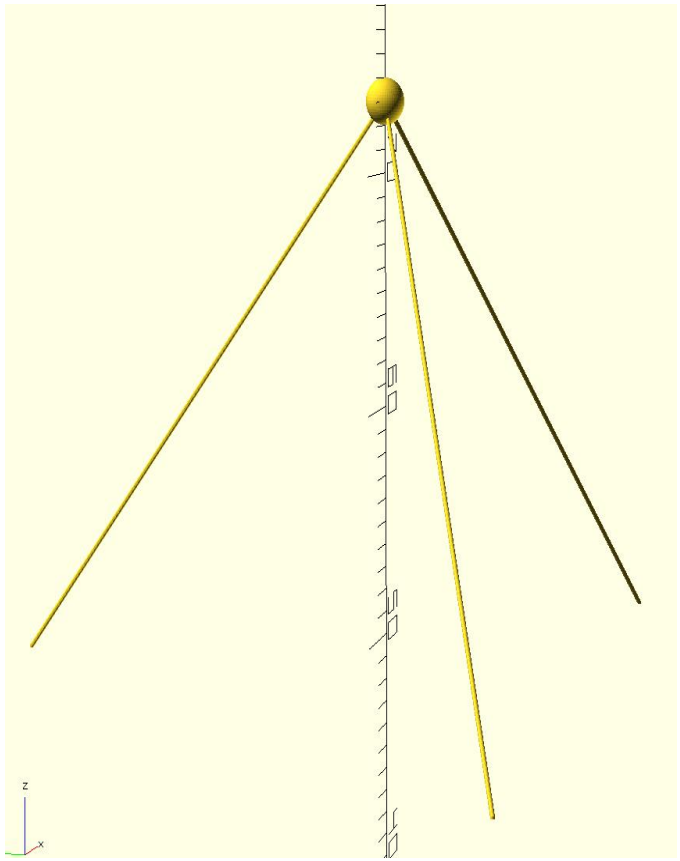
Using hull() method to link 3 single cylinders with a cylinder at the top of them.

Then we get this!



Finally, we generate a sphere at the top.





(the result)

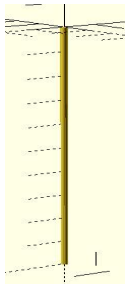
3. Lower part

3 parts: rope + sphere + cone

1.Rope

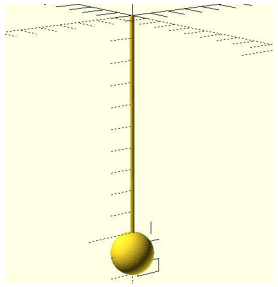
Generate a circle, with small radius.

Then using linear-extrude() to make it into a cylinder, with height=10.Translate it under z=0.



2.sphere

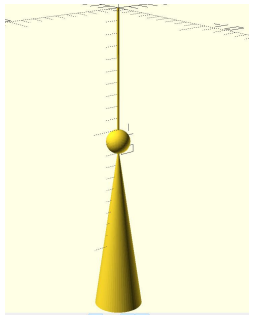
Generate a sphere, with r=1. translate it under the Rope part.

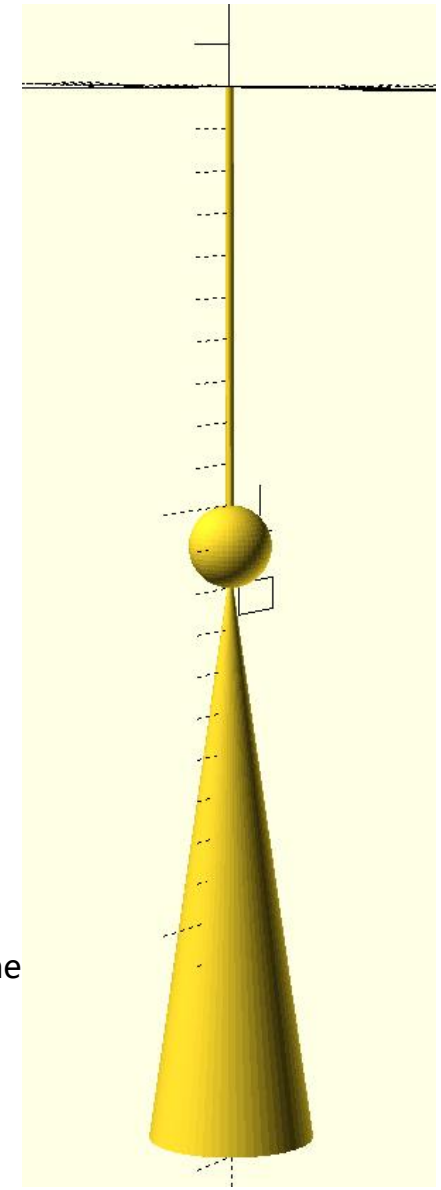


3.Cone

Generate a cylinder with height=14, r1=2, r2=0, making it a cone

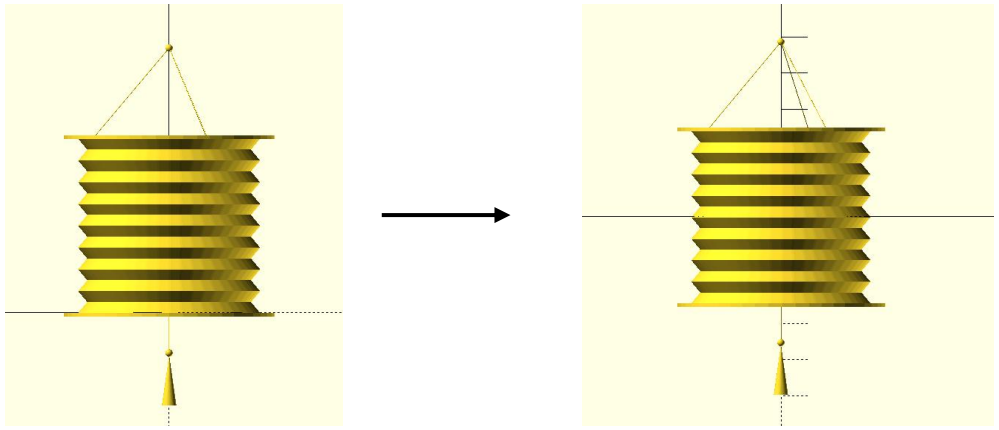
Translate it under sphere.





(the result)

4. Adjusting the position:



The smallest z value of this model is -25.5.
The biggest z value of this model is 74.
The model is 99.5 high; the middle of this model is 49.75.
Then where is this middle point at now? : z=49.75-25.5=24.25
Move the whole model down 24.25, then we get this model with its middle at [0,0,0].

How to rotate the model and get it loaded

A simple summary

Based on the lantern model, we added a rotation command, enhanced the light source, and finally loaded a lantern model that could rotate on the web page.

Implementation steps

-Step 1:

We load the THREE.js library (version r128, compressed version) from the network and load the STLLoader from the network (specifically used for loading.stl format).

```
<body>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/three.js/r128/three.min.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/three@0.128.0/examples/js/loaders/STLLoader.js"></script>
```

We set the width and height (window.innerWidth / window.innerHeight) that the camera sees. Also, we set the camera position at z=100.

```
const scene = new THREE.Scene();
const camera = new THREE.PerspectiveCamera(75, window.innerWidth / window.innerHeight, 0.1, 1000);
camera.position.z = 100;
```

-Step 2:

Add a white light(0xffffff)source at position (10, 10, 10).

```
const light = new THREE.PointLight(0xffffff, 1);
light.position.set(10, 10, 10);
scene.add(light);
```

How to rotate the model and get it loaded

-Step 3:

_Load the lantern model using STLLoader and add it to the scene
Define the variable 'lantern' (for storing the loaded 3D model)
Create a material object (using MeshPhongMaterial),default gray
Set the position of the lantern model(0,0,10)
Set the initial scaling ratio

```
let lantern;
const loader = new THREE.STLLoader();
loader.load('FinalProject.stl', function (geometry) {
  const material = new THREE.MeshPhongMaterial();
  lantern = new THREE.Mesh(geometry, material);
  lantern.position.set(0, 0, 10);
  lantern.scale.set(0.1, 0.1, 0.1);
  scene.add(lantern);
});
```

-Step 4

_The model rotates around the y-axis

```
if (lantern) {
  lantern.rotation.y += 0.01;
```

The model approaches the camera along the Z-axis

```
if (lantern.position.z > 0) lantern.position.z -= 0.05;
```

Model magnification

```
  if (lantern.scale.x < 1) {
    lantern.scale.x += 0.01;
    lantern.scale.y += 0.01;
    lantern.scale.z += 0.01;
  }
}
```

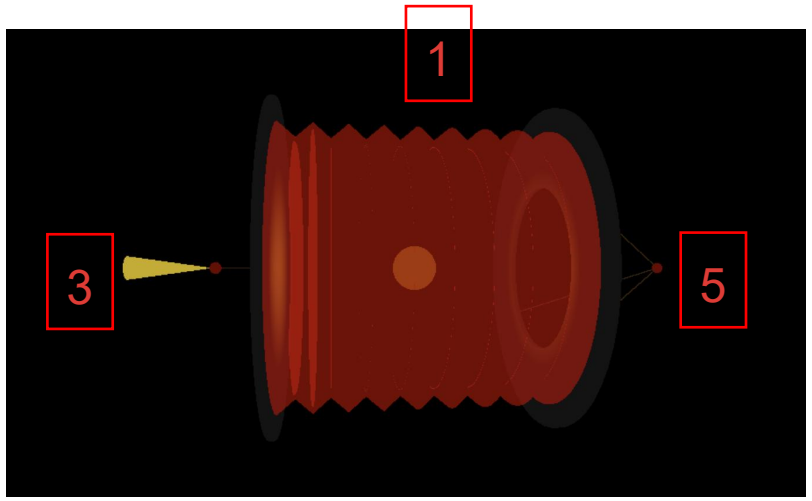
Result

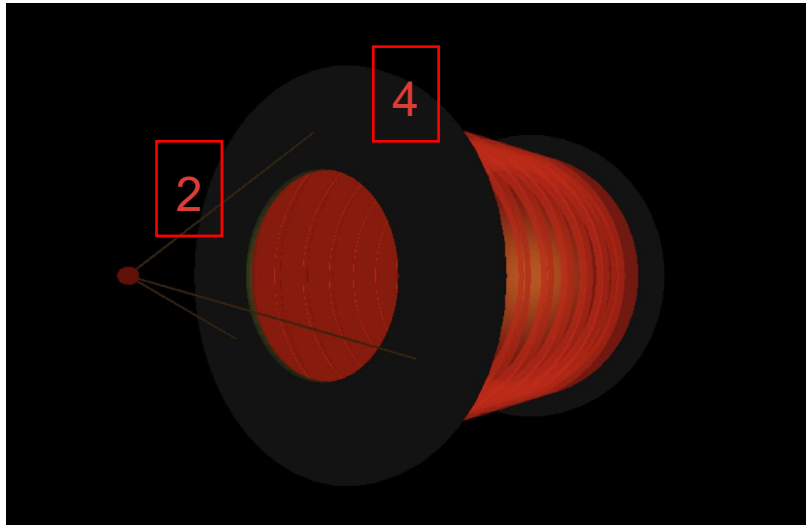
_Finally, start the local server through terminal and run the project on the web page to display.

Definition of Multiple Materials for Different Model Components

In the code, multiple materials are defined for different model components, using different material types from Three.js, such as MeshLambertMaterial, MeshPhongMaterial, and MeshStandardMaterial.

Select different material types according to the requirements of different model components and set corresponding properties such as color, transparency, specular intensity, roughness, and metalness to achieve different appearance effects.






```
// 材質
const mat_zhezhou = new THREE.MeshLambertMaterial({
    color: 0xff0000,
    transparent: true,
    opacity: 0.6
});
const mat_diaosheng = new THREE.MeshLambertMaterial({
    color: 0x4b2e0f,
    transparent: false
});
const mat_liusu = new THREE.MeshPhongMaterial({
    color: 0xffd700,
    shininess: 100,
    specular: 0xffffff
});
const mat_yuanpan = new THREE.MeshStandardMaterial({
    color: 0x222222,
    roughness: 0.6,
    metalness: 0.3
});
const mat_zhuzi = new THREE.MeshPhongMaterial({
    color: 0x8b0000,
    shininess: 80,
    specular: 0xffcccc
});
const mat_zhuziUP = new THREE.MeshPhongMaterial({
    color: 0x8b0000,
    shininess: 80,
    specular: 0xffcccc
});
const mat_liusuxian = new THREE.MeshLambertMaterial({
    color: 0x4b2e0f,
    transparent: false
});
```

```
// 零件配置
const parts = [
    { file: 'model/FinalProject_zhezhou.stl', material: mat_zhezhou, isZhezhou: true },
    { file: 'model/FinalProject_diaosheng.stl', material: mat_diaosheng },
    { file: 'model/FinalProject_liusu.stl', material: mat_liusu },
    { file: 'model/FinalProject_yuanpan.stl', material: mat_yuanpan },
    { file: 'model/FinalProject_zhuzi.stl', material: mat_zhuzi },
    { file: 'model/FinalProject_zhuziUP.stl', material: mat_zhuziUP },
    { file: 'model/FinalProject_liusuxian.stl', material: mat_liusuxian }
];
```

Define multiple material objects, with each material object corresponding to a model component. For example, mat_zhezhou corresponds to the wrinkled part, mat_diaosheng corresponds to the hanging rope part, etc.

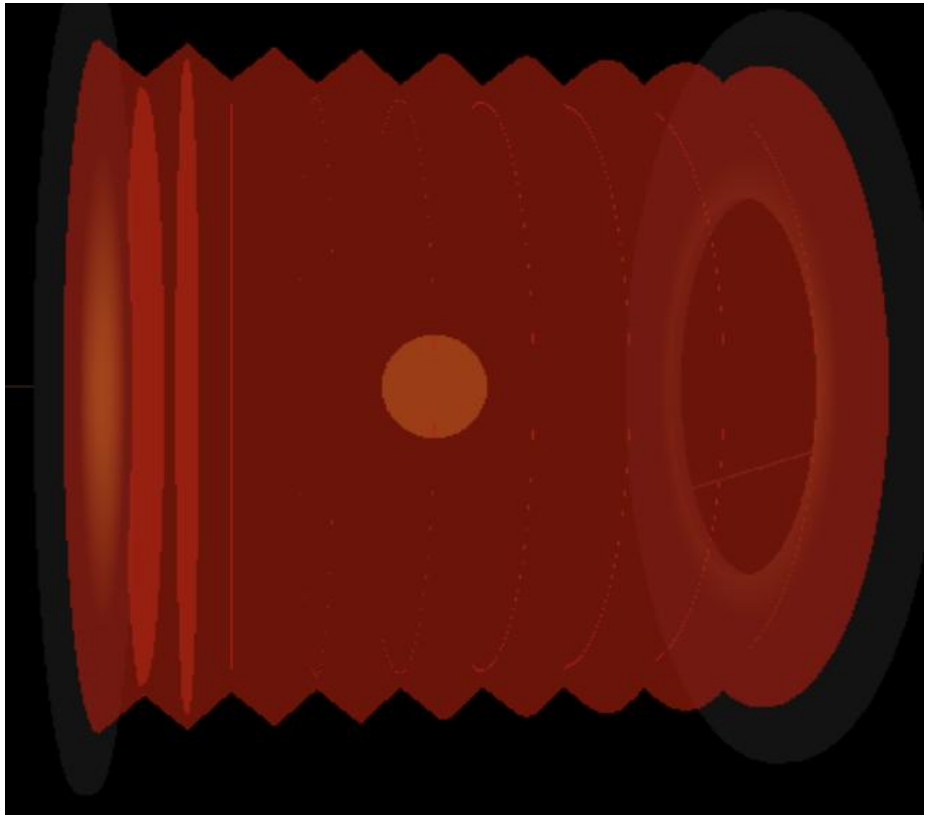
```
// 創建組
const group = new THREE.Group();
scene.add(group);

// STL 加載器
const loader = new THREE.STLLoader();

let meshes = [];

// 加載所有零件
parts.forEach((part, idx) => {
    loader.load(part.file, function (geometry) {
        const mesh = new THREE.Mesh(geometry, part.material);
        mesh.position.set(0, 0, 10);
        mesh.scale.set(0.1, 0.1, 0.1);
        group.add(mesh);
        meshes[idx] = mesh;
    });
});
```

Eg: Simulate the semi - transparent effect of the lantern's paper material.

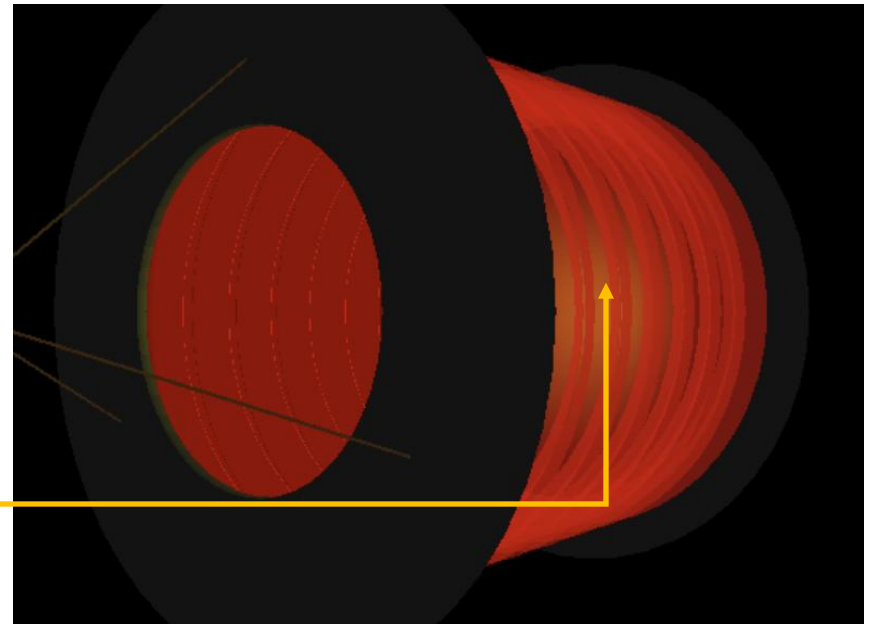


The THREE.MeshLambertMaterial is used to simulate the non - emissive surface of an object like mat_zhezhou, and this surface will have a certain response to lighting.

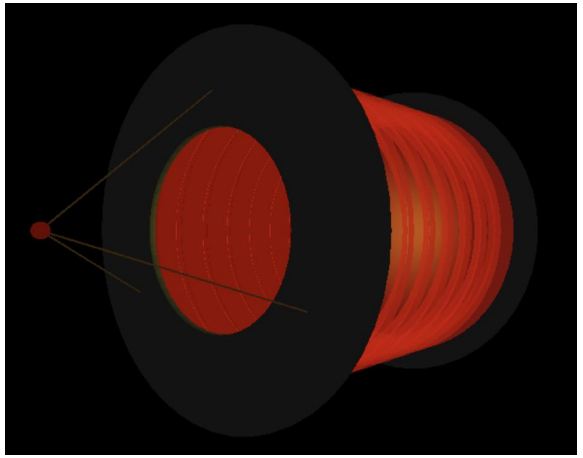
The color property specifies the color as red with a hexadecimal value of 0xff0000, which corresponds to an RGB value of (255, 0, 0).

```
const mat_zhezhou = new THREE.MeshLambertMaterial({
    color: 0xff0000,
    transparent: true,
    opacity: 0.6
});
```

Setting transparent: true indicates that the object should be rendered as semi - transparent. The opacity property ranges from 0.0 (completely transparent) to 1.0 (completely opaque). Here, it is set to 0.6, which represents a translucent paper - like material, allowing objects and the scene behind it to be partially visible.



Lighting Setup

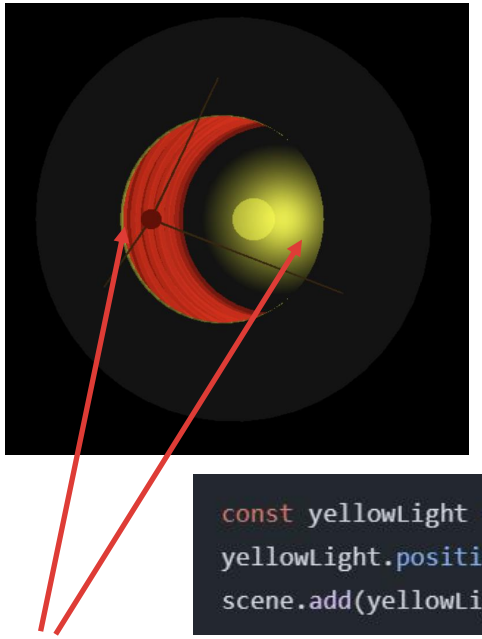


```
const ambient = new THREE.AmbientLight(0xffffff, 0.8);
scene.add(ambient);
```

(1) Ambient Light

In the index.html file, the ambient light is set up using the AmbientLight object from Three.js. Ambient light can uniformly illuminate the entire scene, providing a basic level of brightness.

Create an ambient light object ambient with a white color (0xffffff) and an intensity of 0.8.
Add this ambient light object to the Three.js scene scene so that the scene receives uniform basic illumination.



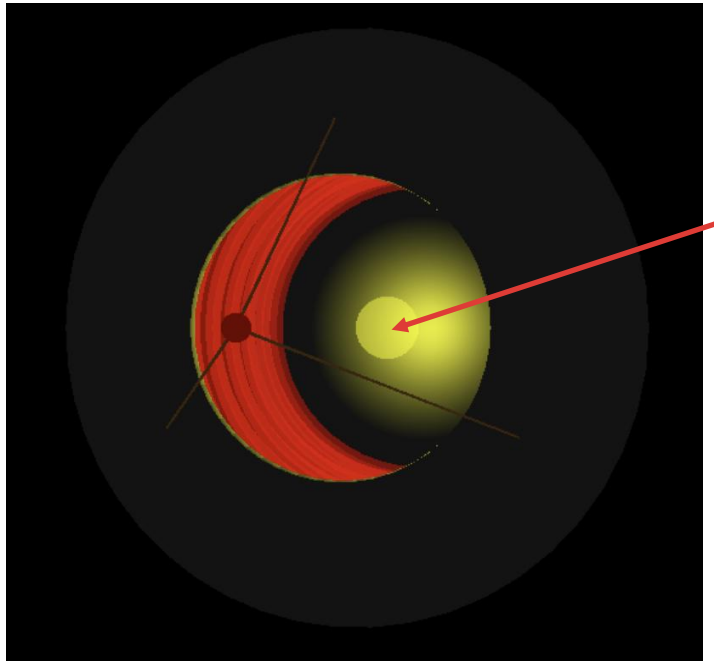
```
const yellowLight = new THREE.PointLight(0xffff00, 20, 30);
yellowLight.position.set(0, 0, 0);
scene.add(yellowLight);
```

(2) Point Light Source

The point light source uses the PointLight object from Three.js.
It can emit light in all directions from a single point, simulating the effect of a light bulb or other point light sources.

Create a yellow (0xffff00) point light source object yellowLight with an intensity of 20 and an effective light range of 30.
Set the position of this point light source at the origin (0, 0, 0).
Add the point light source object to the Three.js scene scene so that there is a point light source illuminating at the origin of the scene.

Lighting Setup



(3) Three - Dimensional Halo Effect

The three - dimensional halo effect is achieved by creating a spherical mesh object and using a semi - transparent material.

```
1  const glowGeometry = new THREE.SphereGeometry(4, 25, 25);
2  const glowMaterial = new THREE.MeshBasicMaterial({
       color: 0xffff00,
       transparent: true,
       opacity: 0.5
   });
3  const glowMesh = new THREE.Mesh(glowGeometry, glowMaterial);
4  glowMesh.position.copy(yellowLight.position);
   scene.add(glowMesh);
5
```

1. Create a spherical geometry object glowGeometry with a radius of 4 and both width and height segments set to 25.
2. Create a basic material object glowMaterial with a yellow color (0xffff00), set it to be semi - transparent (transparent: true), and an opacity of 0.5.
3. Use the spherical geometry and semi - transparent material to create a mesh object glowMesh.
4. Set the position of the mesh object to be the same as that of the point light source yellowLight.
5. Add the mesh object to the Three.js scene scene to create a semi - transparent yellow halo effect at the position of the point light source.

Dynamic camera

Set the parameters of the camera

const radius = 100;// （Set the parameter for the distance between the camera and the center of the model）

Let angle = 0; // （Set the Angle at which the camera observes the model）

const speed = 0.01;// （Set the movement speed of the camera）

Move the camera

angle += speed; （Change the position of the camera）

camer.position.x = radius * Math.cos(angle);

camer.position.z = radius * Math.sin(angle); （Set the movement Angle of the camera to make it move in a circular motion）

camera.lookAt((0,0,10)); （Keep the camera looking at the center of the model while it is moving