CS 460 new assignment: Texture Geometry

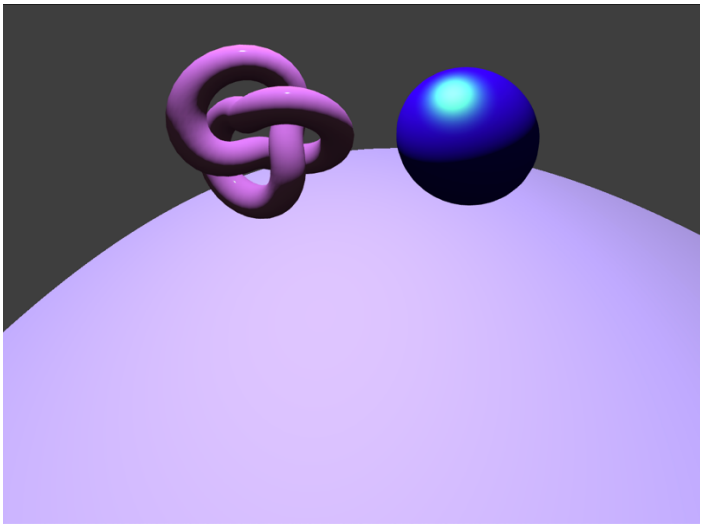
The goal of this assignment is to explore how the material of a 3D object interacts with a spotlight directed towards the object. We will mainly focus on the properties of MeshPhongMaterial and MeshPhysicalMaterial. We have learned that if we want to use mouse drag to rotate the scene, we can use 'OrbitControls(camera, renderer.domElement)'.

Part 1: Create geometry.
We aim to comprehend the parameters of a 'THREE.SphereGeometry.' You can find the relevant information here. Once executed correctly, the sphere geometry should appear on the screen.

Part 2: Create the floor.
Before delving into understanding how the spotlight operates, let's establish the floor. To achieve this, we'll utilize 'THREE.PlaneGeometry.'

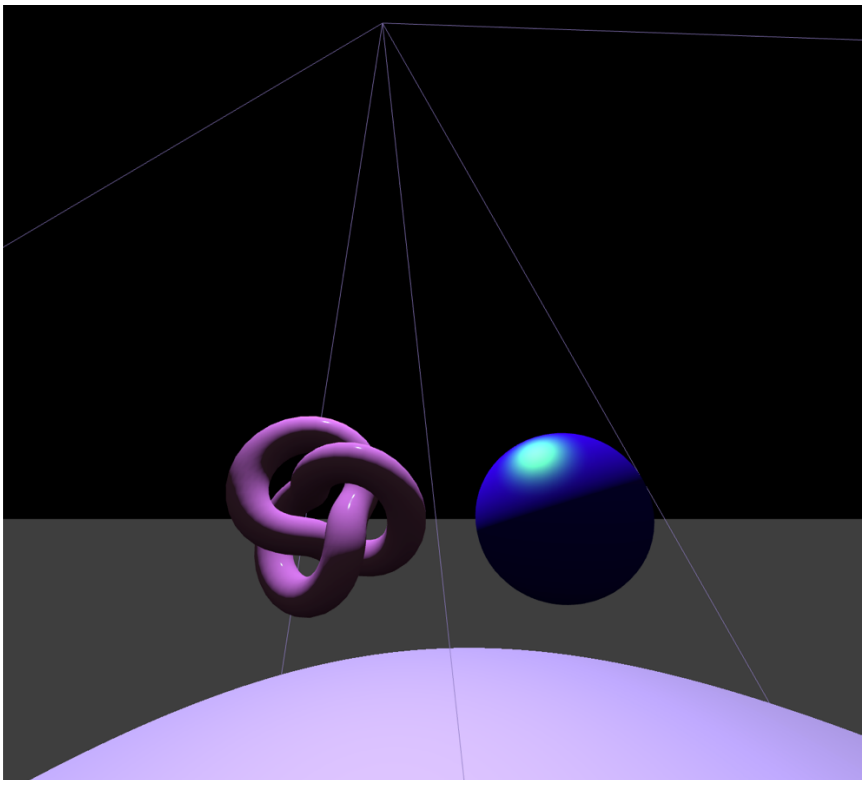
Initially, the plane geometry remains vertical, but for our purposes, it should lie horizontally like a floor. To accomplish this, set the rotation of the plane along the x-axis to $-\pi * 0.5$, ensuring it aligns flatly as depicted below:

Part 3: Set up the spotlight.
To introduce the spotlight onto the screen, use 'THREE.SpotLight.' Afterwards, position the spotlight by utilizing '.position.set(x, y, z);'. Ensure the spotlight is added to the scene to make it visible.



Once the spotlight is added, you can customize its size by utilizing '.angle'. The size adjustment relies on $\frac{\pi}{n}$, where 'n' represents any chosen number. This 'n' value will determine the size of the spotlight. You can learn spotlight properties at https://threejs.org/docs/?q=spotlight#api/en/lights/SpotLight.

Part 4: Add light helper.
To visualize how the spotlight illuminates the plane geometry, utilize 'lightHelper = new THREE.SpotLightHelper(spotLight);'. Remember to include this helper in the scene for proper visualization.



Part 5: Changing property of material.

Previously, you may have utilized 'THREE.MeshBasicMaterial.' However, for this assignment, we'll employ 'THREE.MeshPhongMaterial.' This material allows for the utilization of a non-physically based Blinn-Phong model to calculate reflectance. Information regarding MeshPhongMaterial's properties can be found https://threejs.org/docs/#api/en/materials/MeshPhongMaterial.

In this assignment, our focus will be on the 'shininess' and 'specular' properties. 'shininess' ranges from 0 to 100, while 'specular' denotes the reflected color. These properties can be set as follows:

```
material = new THREE.MeshPhongMaterial({
      color: 0x2F00A7,
      shininess: 30,
      specular: 0x47E800
    });
```

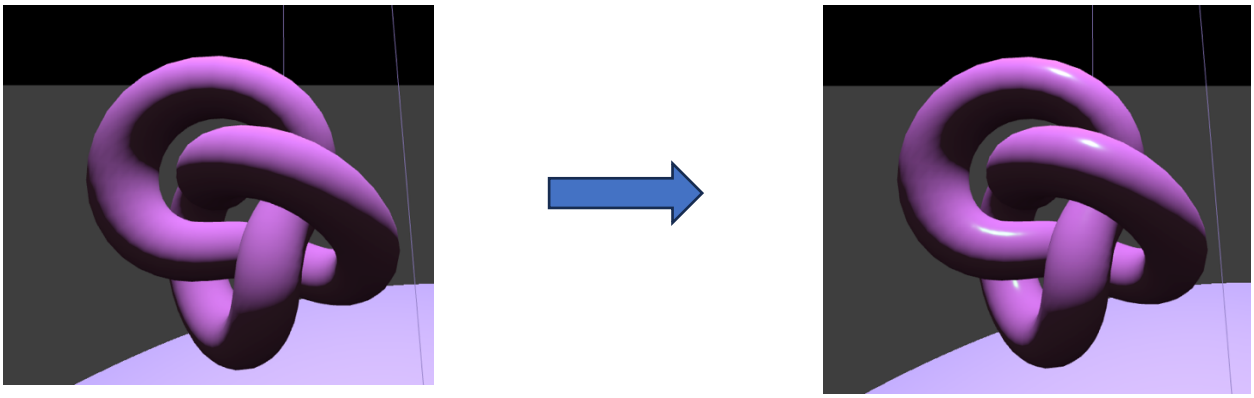
Once these properties are applied to the material, the mesh will appear more realistic.

Part 6: Second mesh with realistic meterial

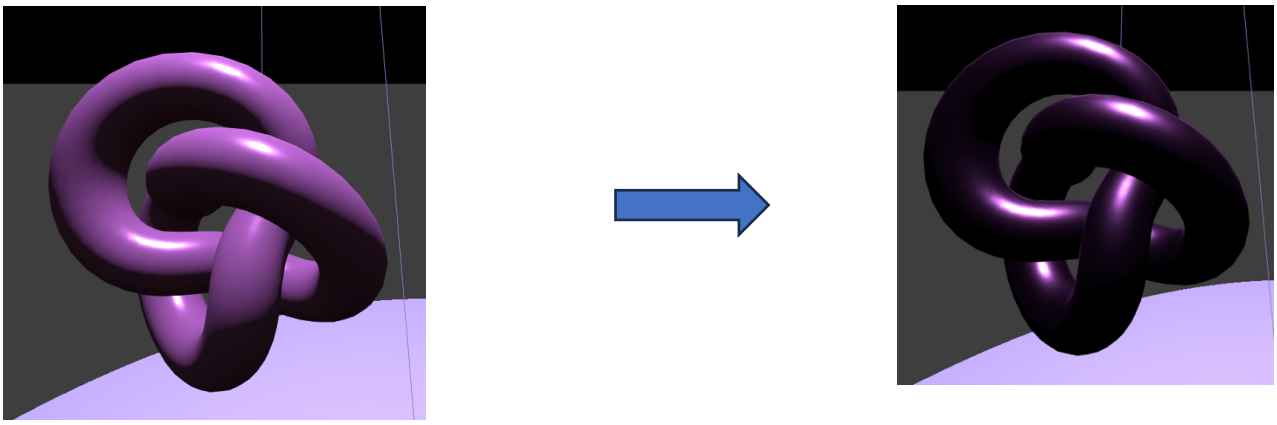
In this assignment, our focus lies on comparing 'MeshPhongMaterial' and 'MeshPhysicalMaterial.' Consequently, we require an additional mesh with a different material. This time, let's utilize 'THREE.TorusKnotGeometry' in conjunction with the 'THREE.MeshPhysicalMaterial.' This setup aims to showcase the disparities between MeshPhongMaterial and MeshPhysicalMaterial.

You can find the properties of MeshPhysicalMaterial here https://threejs.org/docs/?q=mesh#api/en/materials/MeshPhysicalMaterial.sheenRoughness. After you set up the second mesh, we will focus on 3 properties which are ‘metalness’, ‘roughness’, and ‘specularColor’.

‘metalness’ have range from 0.0 to 1.0. 0 is the shiniest 1 is the least.



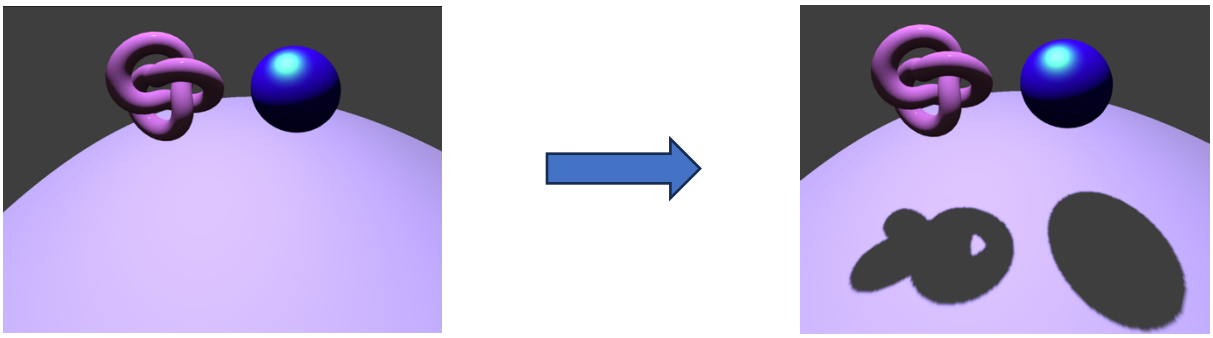
roughness' varies within the range of 0.0 to 1.0. A value of 1.0 will render the mesh with a metallic texture, while a value of 0.0 represents a normal appearance.



Note: When adjusting 'metalness' and 'roughness' together, the mesh reflects a shine similar to real-life metallic surfaces. Additionally, 'specularColor' represents the color that is reflected.

Part 7: Shadow

If you wish the mesh to cast a shadow, you'll need to activate the 'castShadow' property using 'mesh.castShadow = true;'. Additionally, for the plane geometry to display the mesh's shadow, enable its ability to receive shadows with 'mshFloor.receiveShadow = true;'.



Note: Ensure to include 'renderer.shadowMap.enabled = true;' to enable shadow rendering. Omitting this line will result in the absence of shadows.

Part 8: Rotation

Once we've grasped how the mesh reacts to light, our next step involves rotating the spotlight around the mesh. This rotation will entail adjusting the spotlight's position along the x-axis and z-axis. To achieve this, we'll employ mathematical operations. Specifically, the equation for the rotation could be expressed as

$x = \cos(\theta) * radius$ and $z = \sin(\theta) * radius$.

These equations are designed to compute the X and Z positions by leveraging the current time (θ or theta) to generate circular movement. Trigonometric functions, specifically 'cos' and 'sin,' will be utilized to establish this circular motion based on time.

To set θ (theta) as a time variable for animation, use 'var theta = performance.now() / 1000;'. This expression retrieves a high-resolution timestamp in milliseconds and subsequently divides it by 1000 to convert it into seconds.

For the spotlight Movement we will use ‘spotLight.position.set(x, y, z);’ topdates the position of the spotlight to create a circular motion in the x-z plane while maintaining the y-position. Don’t forget ‘lightHelper.update();’ to updates the light helper while the spotlight rotated.

Part 9: Optional Components and GUI Controls

The three.js scene includes optional components that are controlled through a Graphical User Interface (GUI) implemented using the dat.GUI library. These components provide interactive control over various aspects of the scene, allowing for customization and exploration.

Spotlight Control

The GUI provides controls for adjusting the spotlight's properties:

Light Projector:

Allows the selection of different textures (Texture1, Texture2, Texture3, etc.) to be projected by the spotlight. The user can choose from a variety of images to alter the projected light pattern.

Light Intensity:

Enables the user to modify the intensity of the spotlight, controlling the brightness and illumination it provides within the scene.

Spotlight Position (X, Y, Z axes):
Individual sliders for adjusting the position of the spotlight along the X, Y, and Z axes. This allows precise positioning of the spotlight within the scene.
Material Controls
The GUI offers controls to manipulate materials applied to different meshes:

Mesh 1 (MeshPhongMaterial):
Shininess:
Allows adjustment of the shininess attribute, affecting the specular highlight size on the first mesh.

Specular Color:
Enables the user to alter the color of the specular highlights on the first mesh.

Mesh 2 (MeshPhysicalMaterial):
Roughness:
Controls the roughness of the material applied to the second mesh, determining its surface smoothness.

Metalness:
Adjusts the metalness attribute of the material applied to the second mesh, influencing its metallic appearance.

Specular Color:
Allows modification of the specular color attribute, impacting the reflective properties of the second mesh's material.
Here is the code:

```
// GUI
    var gui = new dat.GUI();

    var projector = {
      None: null,
      Texture1: 'wallpaper.png',
      Texture2: 'wallpaper2.png',
      Texture3: 'wallpaper3.png'
      // Add more textures (.png file) here...
    };
    var spotlightPosition = {
      x: spotLight.position.x,
      y: spotLight.position.y,
      z: spotLight.position.z
```

```
};

// GUI for spotlight
var sceneui = gui.addFolder("Spotlight");
sceneui.add(spotLight, 'map', projector).name('Light projector').onChange(function (value) {
  spotLight.map = (value !== 'None') ? new THREE.TextureLoader().load(value) : null;
});
sceneui.add(spotLight, "intensity", 100, 3000).name("Light Intensity");
sceneui.add(spotlightPosition, "x", -20, 20).name("X Axis").onChange(function (value) {
  spotLight.position.x = value;
  lightHelper.update();
});
sceneui.add(spotlightPosition, "y", -20, 20).name("Y Axis").onChange(function (value) {
  spotLight.position.y = value;
  lightHelper.update();
});
sceneui.add(spotlightPosition, "z", -20, 20).name("Z Axis").onChange(function (value) {
  spotLight.position.z = value;
  lightHelper.update();
});
sceneui.open();

// GUI for MeshPhongMaterial
var meshui = gui.addFolder("Mesh1: MeshPhongMaterial");
meshui.add(material, "shininess", 1, 100).name("Shininess");
meshui.addColor(material, "specular").onChange(function () {
    material.color.set(material.color);
  });
meshui.open();

// GUI for MeshPhysicalMaterial
var meshui2 = gui.addFolder("Mesh2: MeshPhysicalMaterial");
meshui2.add(SecondMaterial, "roughness", 0.0, 1.0).name("Roughness");
meshui2.add(SecondMaterial, "metalness", 0.0, 1.0).name("Metalness");
meshui2.addColor(SecondMaterial, "specularColor").onChange(function () {
  SecondMaterial.color.set(SecondMaterial.color);
  });
 meshui2.open();
```


Spotlight
Light projector
None
Light Intensity
1000
X Axis
-2
Y Axis
15
Z Axis
-2
Mesh1: MeshPhongMaterial
Shininess
30
specular
#000000
Mesh2: MeshPhysicalMaterial
Roughness
0
Metalness
0
specularColor
#000000
Close Controls