Add lights are simple, like adding a mesh and adding to the scene. Very simple.

We are going to start by removing the ambient light and the point light from the scene.

Now lets add them back

const ambientLight = new THREE.AmbientLight(0xffffff, .5)

first parm is color, and second is intensity. Now add to scene

const ambientLight = new THREE.AmbientLight(0xffffff, 0.5);

scene.add(ambientLight);

ambient light makes rays form everywhere and is useful in production. Looks like basic mesh. Gives us uniform lighting. So every part of the mesh has basic light on it

can also add this to a debug panel for testing

gui.add(ambientLight, 'intensity').min(0).max(1).step(0.001)

helps to simulate light bouncing

can use light bouncing to get light places, but you can also just apply the ambient light to simulate light bouncing without having to do it.

directionalLight is like the sun

const directionalLight = new THREE.DirectionalLight(0x00fffc, 0.3);

scene.add(directionalLight);

what about the orientation? Well you have to change the position.

The next one is the hemisphere light

const hemisphereLight = new THREE.HemisphereLight(0xff0000, 0x0000ff, 0.5);

scene.add(hemisphereLight);

this makes red come from the top and blue from the bottom. Works like ambient light, every part of mesh is lighted.

Now lets add a point light again

const pointLight = new THREE.pointLight(0xff9000, 0.5);

scene.add(pointLight);

this is just a single light coming from one place or someone using a flashlight. Or light a small lightbulb is in one spot. We can also control and distance and decay on pointlight – looks like campfire

const pointLight = new THREE.PointLight(0xff9000, 0.5, 10, 2);

the third parameter is the distance the light and reach, and the fourth parameter is the decay rate

now lets add the rect area light

const rectAreaLight = new THREE.RectAreaLight(0x4e00ff, 2, 5, 2);

scene.add(rectAreaLight);

strange light that only lights up a rectangle or a plane. Properties are color, intensity, width, and decay.

So imagine a plane of light being shot out. This also only works with meshStandardmaterial, and mesh physical material.

Now lets do the spot light… which is apparently more like a flashlight

const spotLight = new THREE.SpotLight(

  0x78ff00,

  0.5,

  10,

  Math.PI \* 0.1,

  0.25,

  1

);

spotLight.position.set(0, 2, 3);

scene.add(spotLight);

parameters are color, intensity, distance, angle, penumbra, decay

the penumbra how sharp the edges of the light are. With it, or it set at 0, it looks almost animated.

Something odd about the spotlight is that in order to manipulate the target that it shines on, you have to add it to the scene.

spotLight.target.position.x = -0.75;

scene.add(spotLight.target);

looks like the above. This shifts the spotlight slightly to the left

performances, for lights can cost a lot.

You will reach a limit, you can only put up to 50 lights. The idea is to add as few lights as possible. Try to use the ones that cost less.

Minimal cost lights are the Ambient light and the Hemisphere light

Moderate cost lights are the Directional Light and the PointLight

High Cost lights are the Spot Light and the RectAreaLight

Now lets talk about Baking

The idea is to bake the light into the texture. Draw back – can’t move the light then on the texture.

Or helpers that can help us position the light around the scene.

const directionalLightHelper = new THREE.DirectionalLightHelper(

  directionalLight,

  0.2

);

scene.add(directionalLightHelper);

here is an example. The first value is the light you want it to mimic and the second value is the size of the helper.

There is an issue with the spotlight helper and how it does not change If you add the target and move the spotlight

How to get an update to the next frame or update animation again:

Window.requestAnimationFrame(() => {

spotLightHelper.update()

})

This is how you would do it.