# 1.有关光照的知识，参考：[这篇文章](【Three.js入门】灯光与阴影、平行光阴影属性、聚光灯的属性和应用.docx)

# 2.本节示例代码

|  |
| --- |
|  |

## src/lib/threeapp.js

|  |
| --- |
| import \* as THREE from 'three'  import { OrbitControls } from 'three/examples/jsm/controls/OrbitControls'  import Stats from 'three/examples/jsm/libs/stats.module';  //注意：这是第二课的class，内容有点不一样所以我们把class名称改为ThreeApp2  //帧率使用聚光灯而不是平行光  export default class ThreeApp{  constructor(canvasId){  //1.定义场景，相机，渲染器  this.scene = undefined  this.camera = undefined  this.renderer = undefined  //2.定义相机参数,但是保存在本类中  this.fov = 45  this.nearPlane = 1  this.farPlane = 1000  this.canvasId = canvasId  //3.定义额外组件  this.clock = undefined  this.controls = undefined  this.stats = undefined  //4.定义环境光和聚光灯  this.ambientLight = undefined  this.spotLight = undefined  }  initApp(){  //创建场景对象并且赋值给成员变量  this.scene = new THREE.Scene()  //创建相机对象并且用相机成员变量接收  this.camera = new THREE.PerspectiveCamera(  this.fov,  window.innerWidth/window.innerHeight,  this.nearPlane,  this.farPlane  )  //调整相机位置  // this.camera.position.z = 48  this.camera.position.z = 16 //cameraz轴值越大，图像越小  //根据传入的id获取画布对象  let canvas = document.getElementById(this.canvasId)  //创建渲染器  this.renderer = new THREE.WebGLRenderer({  canvas,  antialias:true  })  //开启shadowMap  this.renderer.shadowMap.enabled = true  //设置渲染器的渲染尺寸  this.renderer.setSize(window.innerWidth,window.innerHeight)  //添加到body中  document.body.appendChild(this.renderer.domElement)  //创建时钟，轨道控制器，检测帧数(FPS)的工具  this.clock = new THREE.Clock()  this.controls = new OrbitControls(this.camera,this.renderer.domElement)  this.stats = Stats()  //将检测帧数(FPS)的工具添加到body中  document.body.appendChild(this.stats.domElement)  //创建环境光  this.ambientLight = new THREE.AmbientLight(0xffffff,0.5)  this.ambientLight.castShadow = true  //把环境光添加到场景中  this.scene.add(this.ambientLight)  //创建聚光灯  this.spotLight = new THREE.SpotLight(0xffffff,1)  this.spotLight.castShadow = true //显示阴影  //设置方向光的位置  this.spotLight.position.set(0,64,32) //位置也不一样  //添加方向光到场景中  this.scene.add(this.spotLight)  //给window对象添加事件监听，用来实现窗口的响应式功能  window.addEventListener('resize',()=>this.onWindowResize(),false)  }    //定义animate成员函数，这个函数在外部调用  animate(){  window.requestAnimationFrame(this.animate.bind(this))  this.render()  this.stats.update()  this.controls.update()  }  render(){  this.renderer.render(this.scene,this.camera)  }  onWindowResize(){  //重新计算相机的宽高比  this.camera.aspect = window.innerWidth/window.innerHeight  //更新相机的投影矩阵  this.camera.updateProjectionMatrix()  //重新设置渲染器的渲染大小  this.renderer.setSize(window.innerWidth,window.innerHeight)  }  } |

## src/App.js

|  |
| --- |
| import { useEffect } from 'react'  import './App.css'  import \* as THREE from 'three'  import { GUI } from 'three/examples/jsm/libs/lil-gui.module.min'  import ThreeApp from './lib/threeapp'  // import ThreeApp from './lib/threeapp2'  function main1() {  //创建three应用程序对象  let threeApp = new ThreeApp("myThreeJSCanvas")  //初始化场景  threeApp.initApp()  //调用动画渲染效果  threeApp.animate()  //创建GUI对象  let gui = new GUI()  //创建mainGroup  let mainGroup = new THREE.Group()  mainGroup.position.y = 0.5  threeApp.scene.add(mainGroup) //把组添加到场景中  //设置地面  let groundGeometry = new THREE.BoxGeometry(8, 0.5, 8)  let groundMaterial = new THREE.MeshPhongMaterial({ color: 0xfafafa })  let ground = new THREE.Mesh(groundGeometry, groundMaterial)  ground.receiveShadow = true  ground.position.y = -2  //把地面添加到mainGroup  mainGroup.add(ground)  let boxGeometry = new THREE.BoxGeometry(1, 1, 1)  let boxMaterial = new THREE.MeshPhongMaterial({ color: 0xff0000 })  let box1 = new THREE.Mesh(boxGeometry, boxMaterial)  box1.position.x = -2  box1.castShadow = true  mainGroup.add(box1)  let box2Geometry = new THREE.BoxGeometry(1, 1, 1)  let box2Material = new THREE.MeshPhongMaterial({ color: 0x00ff00 })  let box2 = new THREE.Mesh(box2Geometry, box2Material)  box2.position.x = 0  box2.castShadow = true  mainGroup.add(box2)  let box3Geometry = new THREE.BoxGeometry(1, 1, 1)  let box3Material = new THREE.MeshPhongMaterial({ color: 0x0000ff })  let box3 = new THREE.Mesh(box3Geometry, box3Material)  box3.position.x = 2  box3.castShadow = true  mainGroup.add(box3)  // set up ambient light  let al = new THREE.AmbientLight(0xffffff, 0.5)  mainGroup.add(al)  // set up ambient light gui  let alFolder = gui.addFolder('ambient lignt')  let alSetting = { color: al.color.getHex() }  alFolder.add(al, 'visible')  alFolder.add(al, 'intensity', 0, 1, 0.1)  alFolder.addColor(alSetting, 'color').onChange((value) => al.color.set(value))  alFolder.open()  // setup directional light  let dl = new THREE.DirectionalLight(0xff0000, 0.5)  dl.position.set(0, 2, 0)  dl.castShadow = true  mainGroup.add(dl)  // set up directional light gui  let dlFolder = gui.addFolder("directional light")  let dlSetting = {  visible: true,  color: dl.color.getHex()  }  dlFolder.add(dlSetting, 'visible').onChange((value) => dl.visible = value)  dlFolder.add(dl, 'intensity', 0, 1, 0.25)  dlFolder.add(dl.position, 'y', 1, 4, 0.5)  dlFolder.add(dl, 'castShadow')  dlFolder.addColor(dlSetting, 'color').onChange((value) => dl.color.set(value))  dlFolder.open()  // set up spot light + helper  let sl = new THREE.SpotLight(0x00ff00, 1, 8, Math.PI / 8, 0)  sl.position.set(0, 2, 2)  let slHelper = new THREE.SpotLightHelper(sl)  mainGroup.add(sl, slHelper)  // set up spot light gui  let slFolder = gui.addFolder('spotlight')  let slSettings = {  visible: true  }  slFolder.add(slSettings, 'visible').onChange((value) => {  sl.visible = value  slHelper.visible = value  })  slFolder.add(sl, 'intensity', 0, 4, 0.5)  slFolder.add(sl, 'angle', Math.PI / 16, Math.PI / 2, Math.PI / 16)  slFolder.add(sl, 'castShadow')  slFolder.open()  //场景点光源  let pl = new THREE.PointLight(0xffffff, 1, 8, 2)  pl.position.set(2, 2, 2)  let plHelper = new THREE.PointLightHelper(pl, 0.5)  mainGroup.add(pl, plHelper)  //setup point lignt gui  let plFolder = gui.addFolder('point light')  let plSettings = {  visible: true,  color: pl.color.getHex()  }  plFolder.add(plSettings, 'visible').onChange((value) => {  pl.visible = value,  plHelper.visible = value  })  plFolder.add(pl, 'intensity', 0, 2, 0.25)  plFolder.add(pl.position, 'x', -2, 4, 0.5)  plFolder.add(pl.position, 'y', -2, 4, 0.5)  plFolder.add(pl.position, 'z', -2, 4, 0.5)  plFolder.add(pl, 'castShadow')  plFolder.addColor(plSettings, 'color').onChange((value) => pl.color.set(value))  plFolder.open();  () => {  return gui.destroy()  }  }  function main2() {  //创建three应用程序对象  let threeApp = new ThreeApp("myThreeJSCanvas")  //初始化场景  threeApp.initApp()  //调用动画渲染效果  threeApp.animate()  //创建GUI对象  let gui = new GUI()  //创建mainGroup  let mainGroup = new THREE.Group()  mainGroup.position.y = 0.5  threeApp.scene.add(mainGroup) //把组添加到场景中  // normal box  const bg0 = new THREE.BoxGeometry(1, 1, 1);  const bm0 = new THREE.MeshNormalMaterial();  const box0 = new THREE.Mesh(bg0, bm0);  threeApp.scene.add(box0);  //圆环体  const bgt = new THREE.TorusGeometry(1.5, 0.75, 64, 64);  const bmt = new THREE.MeshNormalMaterial({ color: 0xff0000 });  const torus = new THREE.Mesh(bgt, bmt);  torus.castShadow = true;  torus.position.y = 1;  torus.position.z = 1;  torus.rotation.x = -Math.PI / 3;  mainGroup.add(torus);  //设置地面  let groundGeometry = new THREE.BoxGeometry(8, 0.5, 8)  let groundMaterial = new THREE.MeshPhongMaterial({ color: 0xfafafa })  let ground = new THREE.Mesh(groundGeometry, groundMaterial)  ground.receiveShadow = true  ground.position.y = -2  //把地面添加到mainGroup  mainGroup.add(ground)  let boxGeometry = new THREE.BoxGeometry(1, 1, 1)  let boxMaterial = new THREE.MeshPhongMaterial({ color: 0xff0000 })  let box1 = new THREE.Mesh(boxGeometry, boxMaterial)  box1.position.x = -2  box1.castShadow = true  mainGroup.add(box1)  let box2Geometry = new THREE.BoxGeometry(1, 1, 1)  let box2Material = new THREE.MeshPhongMaterial({ color: 0x00ff00 })  let box2 = new THREE.Mesh(box2Geometry, box2Material)  box2.position.x = 0  box2.castShadow = true  mainGroup.add(box2)  let box3Geometry = new THREE.BoxGeometry(1, 1, 1)  let box3Material = new THREE.MeshPhongMaterial({ color: 0x0000ff })  let box3 = new THREE.Mesh(box3Geometry, box3Material)  box3.position.x = 2  box3.castShadow = true  mainGroup.add(box3)  // set up ambient light  let al = new THREE.AmbientLight(0xffffff, 0.5)  mainGroup.add(al)  // set up ambient light gui  let alFolder = gui.addFolder('ambient lignt')  let alSetting = { color: al.color.getHex() }  alFolder.add(al, 'visible')  alFolder.add(al, 'intensity', 0, 1, 0.25)  alFolder.addColor(alSetting, 'color').onChange((value) => al.color.set(value))  alFolder.open()  // setup directional light  let dl = new THREE.DirectionalLight(0xff0000, 0.5)  dl.position.set(0, 2, 0)  dl.castShadow = true  const dlHelper = new THREE.DirectionalLightHelper(dl, 3);  mainGroup.add(dl, dlHelper)  // set up directional light gui  let dlFolder = gui.addFolder("directional light")  let dlSetting = {  visible: true,  color: dl.color.getHex()  }  dlFolder.add(dlSetting, 'visible').onChange((value) => {  dl.visible = value,  dlHelper.visible = value  })  dlFolder.add(dl, 'intensity', 0, 1, 0.25)  dlFolder.add(dl.position, 'y', 1, 4, 0.5)  dlFolder.add(dl, 'castShadow')  dlFolder.addColor(dlSetting, 'color').onChange((value) => dl.color.set(value))  dlFolder.open()  // set up spot light + helper  let sl = new THREE.SpotLight(0x00ff00, 1, 8, Math.PI / 8, 0)  sl.position.set(0, 2, 2)  let slHelper = new THREE.SpotLightHelper(sl)  mainGroup.add(sl, slHelper)  // set up spot light gui  let slFolder = gui.addFolder('spotlight')  let slSettings = {  visible: true  }  slFolder.add(slSettings, 'visible').onChange((value) => {  sl.visible = value  slHelper.visible = value  })  slFolder.add(sl, 'intensity', 0, 4, 0.5)  slFolder.add(sl, 'angle', Math.PI / 16, Math.PI / 2, Math.PI / 16)  slFolder.add(sl, 'castShadow')  slFolder.open()  //场景点光源  let pl = new THREE.PointLight(0xffffff, 1, 8, 2)  pl.position.set(2, 2, 2)  let plHelper = new THREE.PointLightHelper(pl, 0.5)  mainGroup.add(pl, plHelper)  //setup point lignt gui  let plFolder = gui.addFolder('point light')  let plSettings = {  visible: true,  color: pl.color.getHex()  }  plFolder.add(plSettings, 'visible').onChange((value) => {  pl.visible = value,  plHelper.visible = value  })  plFolder.add(pl, 'intensity', 0, 2, 0.25)  plFolder.add(pl.position, 'x', -2, 4, 0.5)  plFolder.add(pl.position, 'y', -2, 4, 0.5)  plFolder.add(pl.position, 'z', -2, 4, 0.5)  plFolder.add(pl, 'castShadow')  plFolder.addColor(plSettings, 'color').onChange((value) => pl.color.set(value))  plFolder.open();  () => {  return gui.destroy()  }  }  function App() {  useEffect(() => {  main1()  // main2()  }, [])  return (  <>  <div>  <canvas id="myThreeJSCanvas"></canvas>  </div>  </>  )  }  export default App |

# 本示例演示了4种能够产生阴影的灯光的效果，开源慢慢调整参数来观察效果