# 本次场景规划

草坪

雨景

下雨声

树林

跳动的蓝火

阳光

水（three.js内置water.js实现）

天空（three.js内置sky.js实现）

水与阳光以及天空的交互

Three.js内置的water.js可以通过new THREE.Water 获取，除了控制水的透明度，它还可以控制sunDirection sunColor 日照方向、颜色等，distortionScale 是通过噪声来控制水面的UV的波动效果，水面与太阳（天空）的操作通过three.js的sky.js

通过new THREE.Sky()访问，天空的辉度、光长通过turbidity， luminance等访问，水面与光的交互（仅平行光，平行光模拟日光），通过sunDirection: Sunlight.position.clone().normalize(),访问

updateSun 模拟日光的改变（通过改变光照position），flow函数模拟水面流动。

整个水面和灯光有很好的交互同时也可以表现反射

代码：

天空

    var sky = new THREE.Sky();

    sky.scale.setScalar(1000);

    scene.add(sky);

    var uniforms = sky.material.uniforms;

    uniforms.turbidity.value = 1.3;

    uniforms.rayleigh.value = 2.5;

    uniforms.luminance.value = 0.9;

    uniforms.mieCoefficient.value = 0.005;

    uniforms.mieDirectionalG.value = 0.8;

    var parameters = {

      distance: 100,

      inclination: 0.39,

      azimuth: 0.305

    };

水

var waterGeometry = new THREE.PlaneBufferGeometry(1000, 1000);

    water = new THREE.Water(

      waterGeometry, {

        textureWidth: 512,

        textureHeight: 512,

        waterNormals: new THREE.TextureLoader().load('waternormals.jpg', function (texture) {

          texture.wrapS = texture.wrapT = THREE.RepeatWrapping;

        }),

        alpha: 0.9,

        sunDirection: Sunlight.position.clone().normalize(),

        sunColor: 0xffffff,

        waterColor: 0x001e0f,

        distortionScale: 3.7,

        fog: scene.fog !== undefined,

      }

    );

    water.rotation.x = -Math.PI / 2;

    water.material.depthTest = false;

    water.position.set(0, 0.1, 0);

    scene.add(water);

更新阳光照射

    function updateSun() {

      var theta = Math.PI \* (parameters.inclination - 0.5);

      var phi = 2 \* Math.PI \* (parameters.azimuth - 0.5);

      Sunlight.position.x = parameters.distance \* Math.cos(phi);

      Sunlight.position.y = parameters.distance \* Math.sin(phi) \* Math.sin(theta);

      Sunlight.position.z = parameters.distance \* Math.sin(phi) \* Math.cos(theta);

      sky.material.uniforms.sunPosition.value = Sunlight.position.copy(Sunlight.position);

      water.material.uniforms.sunDirection.value.copy(Sunlight.position).normalize();

}

更新水面流动

*//水面流动函数*

    function flow(time) {

      water.material.uniforms.time.value += 1.0 / 60.0;

    }

# 草坪

运用了漫反射贴图，法线、凹凸、光照和高光贴图

     var geometryGrass = new THREE.PlaneBufferGeometry(1000, 1000); *//矩形平面*

*// 加载草地纹理贴图*

    var DifftextureGrass = new THREE.TextureLoader().load("GrassMap/grassmud/brown\_mud\_leaves\_01\_diff\_1k.png");

*// 设置纹理的重复模式*

    DifftextureGrass.wrapS = THREE.RepeatWrapping;

    DifftextureGrass.wrapT = THREE.RepeatWrapping;

*// uv两个方向纹理重复数量*

    DifftextureGrass.repeat.set(10, 10);

    var NormtextureGrass = new THREE.TextureLoader().load("GrassMap/grassmud/brown\_mud\_leaves\_01\_Nor\_1k.png");

*// 设置纹理的重复模式*

    NormtextureGrass.wrapS = THREE.RepeatWrapping;

    NormtextureGrass.wrapT = THREE.RepeatWrapping;

*// uv两个方向纹理重复数量*

    NormtextureGrass.repeat.set(10, 10);

    var BumtextureGrass = new THREE.TextureLoader().load("GrassMap/grassmud/brown\_mud\_leaves\_01\_bump\_1k.png");

*// 设置纹理的重复模式*

    BumtextureGrass.wrapS = THREE.RepeatWrapping;

    BumtextureGrass.wrapT = THREE.RepeatWrapping;

*// uv两个方向纹理重复数量*

    NormtextureGrass.repeat.set(10, 10);

    var LightextureGrass = new THREE.TextureLoader().load("GrassMap/grassmud/brown\_mud\_leaves\_01\_AO\_1k.png");

*// 设置纹理的重复模式*

    LightextureGrass.wrapS = THREE.RepeatWrapping;

    LightextureGrass.wrapT = THREE.RepeatWrapping;

*// uv两个方向纹理重复数量*

    LightextureGrass.repeat.set(10, 10);

    var SeptextureGrass = new THREE.TextureLoader().load("GrassMap/grassmud/brown\_mud\_leaves\_01\_spec\_1k.png");

*// 设置纹理的重复模式*

    SeptextureGrass.wrapS = THREE.RepeatWrapping;

    SeptextureGrass.wrapT = THREE.RepeatWrapping;

*// uv两个方向纹理重复数量*

    SeptextureGrass.repeat.set(10, 10);

    var materialGrass = new THREE.MeshPhongMaterial({

      color: 0xffffff,

      map: DifftextureGrass,

      normalMap: NormtextureGrass,

      normalScale: new THREE.Vector2(1.5, 1.5),

      bumpMap: BumtextureGrass, *//凹凸贴图*

      bumpScale: 3, *//设置凹凸高度，默认值1。*

      lightMap: LightextureGrass, *// 设置光照贴图*

      lightMapIntensity: 1.5, *//烘培光照的强度. 默认 1.*

      specularMap: SeptextureGrass, *//高光贴图*

      shininess: 25,

      side: THREE.DoubleSide, *//两面可见*

*//depthTest: false,*

    });

    var meshGrass = new THREE.Mesh(geometryGrass, materialGrass); *//网格模型对象Mesh*

    scene.add(meshGrass); *//网格模型添加到场景中*

    meshGrass.position.set(0, 0, 0);

    meshGrass.translateY(-0.3);

    meshGrass.rotateX(-Math.PI / 2);

# 雨景

精灵粒子系统

*// 创建一个组表示所有的雨滴*

    var groupRain = new THREE.Group();

*// 批量创建表示雨滴的精灵模型*

*for* (let i = 0; i < 2000; i++) {

      var spriteRainMaterial = new THREE.SpriteMaterial({

        map: textureRain, *//设置精灵纹理贴图*

      });

*// 创建精灵模型对象*

      var spriteDrop = new THREE.Sprite(spriteRainMaterial);

      groupRain.add(spriteDrop);

*// 控制精灵大小,*

      spriteDrop.scale.set(1, 5, 1); *//// 只需要设置x、y两个分量就可以*

      var k1 = Math.random() - 0.5;

      var k2 = Math.random() - 0.5;

*// 设置精灵模型位置，在空间中随机分布*

      spriteDrop.position.set(1000 \* k1, 300 \* Math.random(), 1000 \* k2)

    }

    scene.add(groupRain); *//雨滴群组插入场景中*

*//下雨效果函数*

    function rain() {

*// 每次渲染遍历雨滴群组，刷新频率30~60FPS，两帧时间间隔16.67ms~33.33ms*

*// 每次渲染都会更新雨滴的位置，进而产生动画效果*

      groupRain.children.forEach(sprite => {

*// 雨滴的y坐标每次减2*

        sprite.position.y -= 2;

*if* (sprite.position.y < 0) {

*// 如果雨滴落到地面，重置y，从新下落*

          sprite.position.y = 300;

        }

      });

    }

# 跳动的蓝火

精灵粒子系统 TWEEN.js

generateFireSprite 更新火焰粒子，同时可以控制渐变颜色

initSparkParticle 控制粒子的运动

Tween.js 控制整体动画效果

*//火焰粒子*

*//refer:https://www.jianshu.com/p/1d8666399b8f*

    function generateFireSprite() {

      var canvas = document.createElement('canvas');

      canvas.width = 16;

      canvas.height = 16;

      var context = canvas.getContext('2d');

      var gradient = context.createRadialGradient(canvas.width / 2, canvas.height / 2, 0, canvas.width / 2, canvas

        .height / 2, canvas.width / 2);

      gradient.addColorStop(0, 'rgba(255,255,255,1)');

      gradient.addColorStop(0.2, 'rgba(0,255,255,1)');

      gradient.addColorStop(0.4, 'rgba(0,0,64,1)');

      gradient.addColorStop(1, 'rgba(0,0,0,1)');

      context.fillStyle = gradient;

      context.fillRect(0, 0, canvas.width, canvas.height);

*return* canvas;

    }

    function initSparkParticle(particle, delay) {

      var particle = *this* instanceof THREE.Sprite ? *this* : particle;

      var delay = delay !== undefined ? delay : 0;

      particle.position.set(0, 0, 0);

      particle.scale.x = particle.scale.y = Math.random() \* 32 + 16;

      new TWEEN.Tween(particle)

        .delay(delay)

        .to({}, 400)

        .onComplete(initSparkParticle)

        .start();

      new TWEEN.Tween(particle.position)

        .delay(delay)

        .to({

*//控制火焰高度*

*//x: Math.random() \* 3000 - 500,*

          y: Math.random() \* 3000 - 500,

*//z: Math.random() \* 3000 - 500*

        }, 10000)

        .start();

      new TWEEN.Tween(particle.scale)

        .delay(delay)

        .to({

          x: 0.01,

          y: 0.01

        }, 1000)

        .start();

    }

    function Fire() {

      TWEEN.update(); *//更新火苗*

    }

# 树林

精灵模型，使树面片【贴图】永远正对相机（用户视角）达到三维效果

*/\*\**

*\* 精灵创建树林效果*

*\*/*

*// 加载树纹理贴图*

    var textureRain = new THREE.TextureLoader().load("tree.png");

*// 批量创建表示一个树的精灵模型*

*for* (let i = 0; i < 20; i++) {

      var spriteMaterialTree = new THREE.SpriteMaterial({

        map: textureRain, *//设置精灵纹理贴图*

*//depthTest:false,*

      });

*// 创建精灵模型对象*

      var spriteTree = new THREE.Sprite(spriteMaterialTree);

      scene.add(spriteTree);

*// 控制精灵大小,*

      spriteTree.scale.set(100, 100, 1); *//// 只需要设置x、y两个分量就可以*

      var k1 = Math.random() - 0.5;

      var k2 = Math.random() - 0.5;

*// 设置精灵模型位置，在xoz平面上随机分布*

      spriteTree.position.set(1000 \* k1, 50, 1000 \* k2)

    }

# 远景

天空盒材质为双向

*/\**

*添加天空盒*

*\*/*

    var Skybox = new THREE.SphereGeometry(600, 50, 50);

    var materialSkybox = new THREE.MeshBasicMaterial({

      color: 0xffffff,

      side: THREE.DoubleSide,

    });

    var meshSkybox = new THREE.Mesh(Skybox, materialSkybox);

    scene.add(meshSkybox);

    var textureLoader = new THREE.TextureLoader();

    var texture = textureLoader.load('satara\_night\_1k.jpg');

    meshSkybox.material.map = texture;

    scene.background = texture;

# 下雨声

音频源循环播放

*/\**

*添加音频源*

*\*/*

    var listener = new THREE.AudioListener();

    var audio = new THREE.Audio(listener);

    var audioLoader = new THREE.AudioLoader();

    audioLoader.load('Rain.mp3', function (AudioBuffer) {

      audio.setBuffer(AudioBuffer);

      audio.setLoop(true);

      audio.setVolume(0.3);

      audio.play()

    });