First we need to add a group for the house:

const house = new THREE.Group()

scene.add(house);

group is just an object

Now lets create the walls.

//create a box for the house instead of 4 walls

const walls = new THREE.Mesh(

  new THREE.BoxBufferGeometry(4, 2.5, 4),

  new THREE.MeshStandardMaterial({ color: "#ac8e82" })

);

walls.position.y = 2.5 \* 0.5;

house.add(walls);

now we add the roof

const roof = new THREE.Mesh(

  new THREE.ConeBufferGeometry(3.5, 1, 4),

  new THREE.MeshStandardMaterial({ color: "#b35f45" })

);

roof.position.y = 2.5 + 0.5;

roof.rotation.y = Math.PI \* 0.25;

house.add(roof);

we move the roof up the height of the walls, and then half of its height since its center point is the starting point.

Now we add the door to the house. Here is our door for now:

const door = new THREE.Mesh(

  new THREE.PlaneBufferGeometry(2, 2),

  new THREE.MeshStandardMaterial({ color: "#aa7b7b" })

);

door.position.y = 1;

door.position.z = 2 + 0.01;

house.add(door);

Now we add the bushes to our house:

const bushGeometry = new THREE.SphereBufferGeometry(1, 16, 16);

const bushMaterial = new THREE.MeshStandardMaterial({ color: "#89c854" });

const bush1 = new THREE.Mesh(bushGeometry, bushMaterial);

bush1.scale.set(0.5, 0.5, 0.5);

bush1.position.set(0.8, 0.2, 2.2);

const bush2 = new THREE.Mesh(bushGeometry, bushMaterial);

bush2.scale.set(0.25, 0.25, 0.25);

bush2.position.set(1.4, 0.1, 2.1);

const bush3 = new THREE.Mesh(bushGeometry, bushMaterial);

bush3.scale.set(0.4, 0.4, 0.4);

bush3.position.set(-0.8, 0.1, 2.2);

const bush4 = new THREE.Mesh(bushGeometry, bushMaterial);

bush4.scale.set(0.15, 0.15, 0.15);

bush4.position.set(-1, 0.05, 2.6);

house.add(bush2, bush1, bush3, bush4);

Now we are going to add some graves with a function basically to our haunted house

const graves = new THREE.Group();

scene.add(graves);

add a group for our graves

Add fog this way

//fog

const fog = new THREE.Fog("#262837", 1, 15);

scene.fog = fog;

first param is color, second is where fog starts, and third is when it is at its fullest

Add to around the outskirts of the scene to prevent drop off view

renderer.setClearColor("#262837");

by setting this the same color it removes the edges mostly.

Now lets load all of our textures

const textureLoader = new THREE.TextureLoader();

const doorColorTexture = textureLoader.load("/textures/door/color.jpg");

const doorAlphaTexture = textureLoader.load("/textures/door/alpha.jpg");

const doorAmbientOcclusionTexture = textureLoader.load(

  "/textures/door/ambientOcclusion.jpg"

);

const doorHeightTexture = textureLoader.load("/textures/door/height.jpg");

const doorNormalTexture = textureLoader.load("/textures/door/normal.jpg");

const doorMetalnessTexture = textureLoader.load("/textures/door/metalness.jpg");

const doorRoughnessTexture = textureLoader.load("/textures/door/roughness.jpg");

/\*\*

Here is what the door looks like:

//const door

const door = new THREE.Mesh(

  new THREE.PlaneBufferGeometry(2.2, 2.2, 100, 100),

  new THREE.MeshStandardMaterial({

    map: doorColorTexture,

    alphaMap: doorAlphaTexture,

    transparent: true,

    aoMap: doorAmbientOcclusionTexture,

    displacementMap: doorHeightTexture,

    displacementScale: 0.1,

    normalMap: doorNormalTexture,

    metalnessMap: doorMetalnessTexture,

    roughnessMap: doorRoughnessTexture,

  })

);

door.geometry.setAttribute(

  "uv2",

  new THREE.Float32BufferAttribute(door.geometry.attributes.uv.array, 2)

);

Now, lets add brick to the walls

const brickColorTexture = textureLoader.load("/textures/bricks/color.jpg");

const brickAmbientOcclusionTexture = textureLoader.load(

  "/textures/bricks/ambientOcclusion.jpg"

);

const brickNormalTexture = textureLoader.load("/textures/bricks/normal.jpg");

const brickRoughnessTexture = textureLoader.load(

  "/textures/bricks/roughness.jpg"

);

const walls = new THREE.Mesh(

  new THREE.BoxBufferGeometry(4, 2.5, 4),

  new THREE.MeshStandardMaterial({

    map: brickColorTexture,

    aoMap: brickAmbientOcclusionTexture,

    normalMap: brickNormalTexture,

    roughnessMap: brickRoughnessTexture,

  })

);

walls.geometry.setAttribute(

  "uv2",

  new THREE.Float32BufferAttribute(walls.geometry.attributes.uv.array, 2)

);

Add in grass textures

const graassColorTexture = textureLoader.load("/textures/grass/color.jpg");

const grassAmbientOcclusionTexture = textureLoader.load(

  "/textures/grass/ambientOcclusion.jpg"

);

const grassNormalTexture = textureLoader.load("/textures/grass/normal.jpg");

const grassRoughnessTexture = textureLoader.load(

  "/textures/grass/roughness.jpg"

);

Now add grass to the floor

const grassColorTexture = textureLoader.load("/textures/grass/color.jpg");

const grassAmbientOcclusionTexture = textureLoader.load(

  "/textures/grass/ambientOcclusion.jpg"

);

const grassNormalTexture = textureLoader.load("/textures/grass/normal.jpg");

const grassRoughnessTexture = textureLoader.load(

  "/textures/grass/roughness.jpg"

);

grassColorTexture.repeat.set(8, 8);

grassAmbientOcclusionTexture.repeat.set(8, 8);

grassNormalTexture.repeat.set(8, 8);

grassRoughnessTexture.repeat.set(8, 8);

grassColorTexture.wrapS = THREE.RepeatWrapping;

grassAmbientOcclusionTexture.wrapS = THREE.RepeatWrapping;

grassNormalTexture.wrapS = THREE.RepeatWrapping;

grassRoughnessTexture.wrapS = THREE.RepeatWrapping;

grassColorTexture.wrapT = THREE.RepeatWrapping;

grassAmbientOcclusionTexture.wrapT = THREE.RepeatWrapping;

grassNormalTexture.wrapT = THREE.RepeatWrapping;

grassRoughnessTexture.wrapT = THREE.RepeatWrapping;

so this texture didn’t want to be repeated, so we used this thing called RepeatWrapping to basically make it do it.

Adding grass to floor

const floor = new THREE.Mesh(

  new THREE.PlaneGeometry(20, 20),

  new THREE.MeshStandardMaterial({

    map: grassColorTexture,

    aoMap: grassAmbientOcclusionTexture,

    normalMap: grassNormalTexture,

    roughnessMap: grassRoughnessTexture,

  })

);

floor.geometry.setAttribute(

  "uv2",

  new THREE.Float32BufferAttribute(floor.geometry.attributes.uv.array, 2)

);

Now we add ghosts

 const ghost1Angle = elapsedTime \* 0.5;

  ghost1.position.x = Math.cos(ghost1Angle) \* 4;

  ghost1.position.z = Math.sin(ghost1Angle) \* 4;

  ghost1.position.y = Math.sin(elapsedTime \* 3);

we added this into the tick() function

Now we duplicate for more ghosts (2)

 const ghost2Angle = -elapsedTime \* 0.32;

  ghost2.position.x = Math.cos(ghost2Angle) \* 5;

  ghost2.position.z = Math.sin(ghost2Angle) \* 5;

  ghost2.position.y = Math.sin(elapsedTime \* 4) + Math.sin(elapsedTime \* 2.5);

we added more randomness to the y axes sin which is when it comes from the group and goes back down.

Now lets activate the shadows

//shadows

renderer.shadowMap.enabled = true;

moonLight.castShadow = true;

doorLight.castShadow = true;

ghost1.castShadow = true;

ghost2.castShadow = true;

ghost3.castShadow = true;

walls.castShadow = true;

bush1.castShadow = true;

bush2.castShadow = true;

bush3.castShadow = true;

bush4.castShadow = true;

floor.receiveShadow = true;

also need to do this in the graves loop for them to cast shadows.

Now optimize shadows. We can do this by changing the map size near and far of each shadow

doorLight.shadow.mapSize.width = 256;

doorLight.shadow.mapSize.height = 256;

doorLight.shadow.camera.far = 7;

ghost1.shadow.mapSize.width = 256;

ghost1.shadow.mapSize.height = 256;

ghost1.shadow.camera.far = 7;

ghost2.shadow.mapSize.width = 256;

ghost2.shadow.mapSize.height = 256;

ghost2.shadow.camera.far = 7;

ghost3.shadow.mapSize.width = 256;

ghost3.shadow.mapSize.height = 256;

ghost3.shadow.camera.far = 7;

looks like this

The last thing we can do is change the algo for shadow.maps to a cleaner algo.

renderer.shadowMap.type = THREE.PCFSoftShadowMap;

Now we can go further. Add new objects, and other things for fun