0 – Basic Scene

Objects are rendered with `THREE.MeshBasicMaterial` as wireframes. They're all the same colour.

The light marker can be repositioned at will as it has a `THREE.TransformControls` attached to it.

The controls set the ambient and directional colours, and the light marker colour is updated in `animate()`. This is a brute force method and bad practice as the colour is updated every frame whether it's changed or not.

Note that although changing the directional colour changes the colour of the models in the scene this is due the shared material changing colour and not the light source.

```
function animate() {
  window.requestAnimationFrame(animate);

  cameraControls.update(clock.getDelta());

  control.update();

  lightMarker.material.color.set(directionalLightColour);

  render();
}
```

1 – Basic Scene → Coloured Materials

Disable wireframe.

```
// MATERIALS
matCurrent = new THREE.MeshBasicMaterial({ color: 0xffffff});
```

Set each model to a different colour. When a mesh is initialised the material passed in the constructor is assigned ***not*** copied, so we must assign a different material for each model as desired. Rather than defining a new material we can clone and changes its colour.

```
var polyhedron = new THREE.Mesh(geometry, matCurrent.clone());
```

The colour can be changed in a variety of different ways as shown below.

```
polyhedron.material.color.setHex(0xffffff);
```

```
matCurrent.color = (new THREE.Color(0xff0000));
```

```
matCurrent.color.b = 255;
```

```
polyhedron.material.color.set(0x00ffff);
```

Synchronise the ambient colour and marker colour. Changing the ambient colour will only affect the light marker.

```
function animate() {
  ambientLight.color.set(ambientLightColour);
  lightMarker.material.color.set(ambientLightColour);
```

2 – Coloured Materials → Directional Light

Add a directional light to the scene.

```
// LIGHTS
light = new THREE.DirectionalLight(directionalLightColour, 1.0);
scene.add(light);
```

Change the `THREE.MeshBasicMaterial` to `THREE.MeshLambertMaterial`. Lighting and shading will be calculated at each vertex then averaged across each triangle.

```
// MATERIALS
matCurrent = new THREE.MeshLambertMaterial({ color: 0xffffff });
```

Change light marker colour to emissive to emissive only.

```
lightMarker.material.color.set(0);
lightMarker.material.emissive.set(directionalLightColour);
```

Sync light and light marker position

```
function animate() {
  window.requestAnimationFrame(animate);

  cameraControls.update(clock.getDelta());
  control.update();

  ambientLight.color.set(ambientLightColour);
  lightMarker.material.emissive.set(directionalLightColour);
  light.position.x = lightMarker.position.x;
  light.position.y = lightMarker.position.y;
  light.position.z = lightMarker.position.z;
  light.color.set(directionalLightColour);
```

This is bad practice and unnecessary, so let's make the `light` a child of the `lightMarker`.

Delete

```
~~light.position.x = lightMarker.position.x;~~
~~light.position.y = lightMarker.position.y;~~
~~light.position.z = lightMarker.position.z;~~
```

and

```
function fillScene() {
  ...
  // LIGHTS
  light = new THREE.DirectionalLight(directionalLightColour, 1.0);
  ~~scene.add(light);~~
```

Add the light to the light marker

```
  geometry = new THREE.SphereGeometry(1, 8, 6);
  matCurrent.color.set(ambientLightColour);
  polyhedron = new THREE.Mesh(geometry, matCurrent.clone());
  polyhedron.position.set(-80, 100, 180);
  polyhedron.scale.set(5, 5, 5);
  lightMarker = polyhedron;
  lightMarker.material.color.set(0);
  lightMarker.material.emissive.set(directionalLightColour);

  scene.add(polyhedron);
  lightMarker.add(light);
```

3 – Directional Light → Shadow

It's shadow time.

Turn on `castShadow` for the light.

```
// LIGHTS
ambientLight = new THREE.AmbientLight(ambientLightColour);
scene.add(ambientLight);

light = new THREE.DirectionalLight(directionalLightColour, 1.0);
//light.shadowCameraVisible = true; //uncomment to see the shadow camera
light.castShadow = true;
```

Turn on `castShadow` and `receiveShadow` for meshes. Note that the teapot is not self-shadowing unless both are set.

```
// MODELS
var geometry;
geometry = new THREE.PlaneGeometry(1, 1);
var polyhedron = new THREE.Mesh(geometry, matCurrent.clone());
polyhedron.rotation.set(-Math.PI / 2, 0, 0);
polyhedron.scale.set(1000, 1000, 1000);
polyhedron.material.color.setHex(0xffffff);
polyhedron.receiveShadow = true;
scene.add(polyhedron);

geometry = new THREE.TeapotGeometry(1, 4, 1, 1, 1, 1, 1);
matCurrent.color = (new THREE.Color(0xff0000));
polyhedron = new THREE.Mesh(geometry, matCurrent.clone());
polyhedron.position.set(40, 30, 0);
polyhedron.scale.set(30, 30, 30);
polyhedron.castShadow = true;
scene.add(polyhedron);
```

Set up the renderer for shadow mapping.

```
function init() {
  // RENDERER
  renderer = new THREE.WebGLRenderer({ antialias: true });
  renderer.gammaInput = true;
  renderer.gammaOutput = true;
  renderer.setSize(window.innerWidth, window.innerHeight);
  renderer.setClearColor(0xAAAABB, 1.0);

  renderer.shadowMapEnabled = true;
  renderer.shadowMapSoft = false;

  renderer.shadowCameraNear = 3;
  renderer.shadowCameraFar = 1000;
  renderer.shadowCameraFov = 50;

  renderer.shadowMapBias = 0.0039;
  renderer.shadowMapDarkness = 0.5;
  renderer.shadowMapWidth = 1024;
  renderer.shadowMapHeight = 1024;
```

4 – Shadow → Phong / per pixel lighting

Change the `THREE.MeshLambertMaterial` to `THREE.MeshPhongMaterial`. Lighting and shading will be calculated on a per pixel basis, making it more accurate.

```
// MATERIALS
matCurrent = new THREE.MeshPhongMaterial({ color: 0xffffff });
```

5 – Phong / per pixel lighting → Point Light

Change the directional light to a point light. Note that shadows no longer work, this is not a bug but due to the way shadows are generated.

```
// LIGHTS
ambientLight = new THREE.AmbientLight(ambientLightColour);
scene.add(ambientLight);

light = new THREE.PointLight(directionalLightColour, 1.0);
```

6 – Point Light → Spotlight

Change the point light to a spotlight. Note that shadows are back. ☺

```
function fillScene() {
  scene = new THREE.Scene();
  scene.fog = new THREE.Fog(0x808080, 2000, 4000);

  // LIGHTS
  ambientLight = new THREE.AmbientLight(ambientLightColour);
  scene.add(ambientLight);

  light = new THREE.SpotLight(directionalLightColour, 1.0);
```

A spotlight has a target, by default this is (0, 0, 0). Changing this is a bit convoluted.

```
  lightTarget = new THREE.Object3D();
  lightTarget.position.set(0,0,0);
  scene.add(lightTarget);
  light.target = lightTarget;
```

The following code has the light target track the spotlight, so making it shine directly down.

```
function animate() {
  ...
  lightTarget.position.x = light.position.x;
  lightTarget.position.z = light.position.z;
```