Aula02 – Projeções, Iluminação e Transformações

João Magalhães, 79923 e João Ferreira, 80041

Visualização de Informação, 2021

Mestrado em Engenharia Informática, Universidade de Aveiro

Tipos de câmaras

Neste exemplo começamos por activar a propriedade *wireframe* no material do cubo definido na aula01 e também por remover a respectiva rotação. Posteriormente, definimos uma OrthographicCamera com a zona de visualização entre -3 e 3 para o eixo das abcissas e -4 e 4 para o eixo das ordenadas. Para terminar este exemplo definimos também uma função para que o cubo não se deformasse quando o tamanho da janela fosse alterado.

Orbit Control & Trackball

Neste exemplo, experimentamos os controlos Orbit Control e Trackball. Para utilizar o controlo Orbit Control basta alterar a variável *controls* para new THREE.OrbitControls passando como argumentos a câmara e o renderer. Este tipo de contrologo permite que a câmara orbite à volta de um objecto.

Como sugerido, também experimentamos o Trackball Controls, que é muito semelhante ao Orbit Controls com a diferença que não mantém sempre a câmara “por cima”.

Iluminação e Materiais

Começamos por voltar activar a rotação do cubo e criamos uma *DirectionalLight* na posição 0,5,0 como indicado. A figura seguinte indica a parte do código onde é realizada esta operação.

```
const directionalLight = new THREE.DirectionalLight(  0xffffff, 1.0 );
directionalLight.position.set( 0 , 5 , 0);
scene.add( directionalLight );
```

Para que o objecto interaja com a luz é necessário criar um material do tipo MeshPhongMaterial com as propriedades specular e shininess.

Finalmente, adicionamos uma luz ambiente com o código que nos é fornecido no enunciado.

Sombreamento

Para este exemplo definimos uma esfera (sphereGeometry), que necessita de, no mínimo três argumentos: raio, widthSegments (número de segmentos horizontais) e heightSegments (número de segmentos verticais). Definimos estes parâmetros como 1, 10 e 10, respectivamente.

De seguida, criamos outra esfera semelhante, mas numa posição oposta no eixo X (2.5 e -2.5). Definimos a posição da seguinte maneira:

sphere.position.x= 2.5;

Definimos também as luzes direcional e ambiente tal como no exemplo 2.3. Para além disso, no material das esferas definimos a propriedade flatShading.

Desenvolvemos também a parte opcional, que consistia em programar um material lambertiano, material este que dispersa a luz de igual forma em todas as direções, ignorando o coeficiente de especular e brilho.

Também desenvolvemos outro material de modo a que a esfera ficasse de cor dourada, como é possível observar na figura seguinte.

```
var material = new THREE.MeshPhongMaterial({
    shading: THREE.SmoothShading
});
material.color = new THREE.Color(0.75164,0.60648,0.22648);
material.specular= new THREE.Color(0.628281,0.555802,0.366065);
material.shininess = 0.4 * 256;
```

Para terminar este exemplo, acrescentamos uma luz direccional vermelha e uma azul e também um green spotlight, como é possível observar na figura seguinte.

```
//Luz direcional vermelha
const directionalLightRed = new THREE.DirectionalLight(  0xff0000, 1.0 );
directionalLightRed.position.set( -5 , 0 , 0);
scene.add( directionalLightRed );

//Luz direcional azul
const directionalLightBlue = new THREE.DirectionalLight(  0x0000ff, 1.0 );
directionalLightBlue.position.set( 5 , 0 , 0);
scene.add( directionalLightBlue );

//Green spotlight
const spotLight = new THREE.SpotLight( 0x00ff00 );
spotLight.position.set( 0, 0, -5 );

scene.add(spotLight);
```

Transparência

Para este exercício definimos duas esferas, uma ao lado de cada uma das duas já definidas anteriormente, utilizando o material fornecido no enunciado. Para este material fomos alterando os valores da opacity, sendo este parâmetro um valor entre 0.0 e 1.0 (0.0 – transparente ; 1.0 – opaco).

O material referido poderia seguir a definição evidenciada na figura abaixo.

```
var glassMaterial = new THREE.MeshPhongMaterial( {
    color: 0x222222,
    specular: 0xFFFFFF,
    shininess: 100,
    opacity: 1,
    transparent: true
} )
```

Transformações (escala e rotação)

Neste exemplo começamos por definir um paralelepípedo de tamanho (2,1,4) recorrendo à propriedade scale. Este paralelepípedo na posição (0,0,0). A figura seguinte mostra o processo para a criação deste objecto.

```
const geometry = new THREE.BoxBufferGeometry( 1, 1, 1 );

const material = new THREE.MeshPhongMaterial( {color: 0xff0000, opacity: 0.5, transparent: true} );

//paralelepipedo
const paralel = new THREE.Mesh( geometry, material );
paralel.position.set( 0, 0, 0 );
paralel.scale.x = 2;
paralel.scale.y = 1;
paralel.scale.z = 4;
```

Posteriormente, definimos quatro esferas com raio 0.5, centradas nos vértices inferiores do paralelepípedo. A definição da esfera é exibida na figura abaixo.

```
//esfera1
var geometry1 = new THREE.SphereGeometry(0.5,10,10);
var material1 = new THREE.MeshPhongMaterial({
    color: 0x00ff00
});

var sphere1 = new THREE.Mesh( geometry1, material1 );
sphere1.position.x = 1;
sphere1.position.y = -0.5;
sphere1.position.z = 2;
```

As restantes três esferas são definidas seguindo o mesmo processo, alterando a posição. A lista de posições corresponde a metade dos valores definidos para o paralelepípedo, alterando os sinais conforme a esfera que pretendemos. As posições de cada esfera seriam (1,-0.5,2), (-1,-0.5,2), (1,-0.5,-2) e (-1,-0.5,-2).

Para a construção deste objecto definimos um grupo, adicionando-lhe as quatro esferas e o paralelepípedo. No fim, é necessário adicionar à scene este grupo.

```
const group = new THREE.Group();
group.add( paralel );
group.add( sphere1 );
group.add( sphere2 );
group.add( sphere3 );
group.add( sphere4 );

scene.add( group );
```

Transformações - Rotações

Para este exercício adaptamos o que já tinha sido desenvolvido no código do exercício anterior.

As esferas inicialmente desenvolvidas foram substituídas por cilindros com raio 0.5 e altura 0.2, como é mostrado na figura abaixo.

```
var geometry1 = new THREE.CylinderGeometry(0.5,0.5,0.2,32);

geometry1.rotateZ(-Math.PI * 0.5);

var material1 = new THREE.MeshBasicMaterial({
    color: 0xffff00
});

var cilindro1 = new THREE.Mesh( geometry1, material1 );
cilindro1.position.x = 1;
cilindro1.position.y = -0.5;
cilindro1.position.z = 2;
```

Por fim criamos um objecto que representa um sistema de coordenadas, recorrendo a três cilindros.

```
var geometryVerde = new THREE.CylinderGeometry(0.1,0.1,1,32);

geometryVerde.rotateZ(-Math.PI * 0.5);

var materialVerde = new THREE.MeshBasicMaterial({
    color: 0x00ff00
});


var cilindroX = new THREE.Mesh( geometryVerde, materialVerde );
cilindroX.position.x = -0.5;
cilindroX.position.y = 1.5;
cilindroX.position.z = 0;
```

O cilindro da figura acima representava o eixo Z uma vez que é invocado o método rotateZ. Para os restantes dois eixos X e Y seria necessário invocar o método rotateX e rotateY, respectivamente. O argumento deste métodos encontra-se em radianos.

Estes três cilindros pertencem a um único grupo chamado eixo, como mostra a figura abaixo.

```
const eixo = new THREE.Group();
eixo.add( cilindroX );
eixo.add( cilindroY );
eixo.add( cilindroZ );

scene.add( eixo );
```

Conclusão

Esta aula proporcionou-nos a capacidade de experimentarmos novos tipos de câmeras, como a OrthographicCamera, bem como perceber o seu funcionamento.

Também aprendemos superficialmente dois novos tipos de controlos, como o Orbit Control e o Trackball.

Para além dos temas já referidos, lidamos também com a questão de iluminação, utilizando directional ligth, e os seus parâmetros specular e shininess. Utilizamos também uma luz ambiente, luz vermelha, luz azul e green spotlight.

Ao longo do guião também obtivemo noções básicas sobre esferas, as suas posições em cada um dos três eixos e também sobre os parâmetros do construtor (widthSegments e heightSegments). Para as esferas criadas, utilizamos também o material Lambertiano.

Finalmente, lidamos com as questões de transparência e grupos de objectos.

Em suma, consideramos que esta aula foi útil, maioritariamente, para entendermos como funcionam os eixos e as posições do ambiente gráfico e, simultaneamente, conciliar esse aspecto posicional com questões de iluminação.