Aula 4- Interação/seleção

- Interação com o rato
- Seleção de objetos com raycast
- Interação com câmara
- Texto 3D

4.1 Interação com o rato

Utilize como base um exemplo das últimas aulas por exemplo o exercício 2.3 com o cubo e alguma iluminação. Desative as animações e controlo da câmara (com orbitControl).

O objetivo é permitir controlar a orientação do objeto através da utilização do rato.

Analise o código seguinte e use-o para permitir rodar o cubo usando o rato (considere phi o ângulo de rotação em x e theta em y).

```
var drag = false;
var phi = 0, theta = 0;
var old_x, old_y;

var mouseDown = function (e) {
    drag = true;
    old_x = e.pageX, old_y = e.pageY;
    e.preventDefault();
    return false;
}

var mouseUp = function (e) {
    drag = false;
}

var mouseMove = function (e) {
    if (!drag) return false;
    var dX = e.pageX - old_x,
    dY = e.pageY - old_y;
    theta += dX * 2 * Math.PI / window.innerWidth;
    phi += dY * 2 * Math.PI / window.innerHeight;
    old_x = e.pageX, old_y = e.pageY;
    e.preventDefault();
}

renderer.domElement.addEventListener("mousedown", mouseDown);
renderer.domElement.addEventListener("mouseup", mouseUp);
renderer.domElement.addEventListener("mousemove", mouseMove);
```

4.2 Seleção de objetos

O `three.js` disponibiliza duas classes (`Projector` e `Raycaster`) que podem ser utilizadas para realizar `raycasting` na cena permitindo desta forma intercetar/selecionar objetos.

A classe `Projector` permite a partir de uma coordenada 2D (pixel na imagem/ecrã) e informação da câmara calcular a direção de um raio 3D que indica todos os pontos 3D da cena que são projetados no pixel original. Deve adicionar a ligação ao script no início do código:

```
<script src="js/examples/js/renderers/Projector.js"></script>
```

Por seu lado a classe `Raycaster` permite, dado a posição da câmara e um vetor direção, emitir um raio na cena e determinar que objetos são intersetados pelo mesmo.

Analise o código seguinte, acrescente-o no exemplo anterior e veja o resultado do mesmo.

```
    //mouse event variables
    var projector = new THREE.Projector(),
    mouse_vector = new THREE.Vector3(),
    mouse = { x: 0, y: 0, z: 1 },
    ray = new THREE.Raycaster(new THREE.Vector3(0, 0, 0), new
THREE.Vector3(0, 0, 0)),
    intersects = [];

function onMouseDown(e) {
    //Evita que o evento chame outra função
    e.preventDefault();

    //Começa a tranformação entre coordenadas do rato e three.js
    mouse.x = (e.clientX / window.innerWidth) * 2 - 1;
    mouse.y = -(e.clientY / window.innerHeight) * 2 + 1;

    //Vector 3D que indica a direção do vetor a partir do pixel
    mouse_vector.set(mouse.x, mouse.y, mouse.z);

    //Define um ponto no espaço 3D de acordo com o clique do rato
    mouse_vector.unproject(camera);

    var direction = mouse_vector.sub(camera.position).normalize();

    //Chama o raycaster com a posição da câmara e a direção
    ray.set(camera.position, direction);
    //Verifica se o raio intersetou algum objeto na cena
    intersects = ray.intersectObject(cube);

    if (intersects.length) {
      alert("hit");
    }
}
renderer.domElement.addEventListener('mousedown', onMouseDown);
```

Coloque outro modelo na cena e modifique o código para permitir distinguir em que modelo foi feito a seleção. Pode ainda modificar a cor do objeto selecionado: ao clicar num dos cubos o mesmo fica vermelho por exemplo.

Adicione um controlo do tipo `OrbitControls` ou `TrackballControls` e veja o que ocorre quando os cubos estão alinhados. Garanta que o código permita selecionar os dois cubos quando os mesmos estão alinhados (pode usar o método `IntersectObjects` do `rayCaster` com a variável `scene.children`).

4.3 Controlo da posição da câmara

O método usado anteriormente no ponto 4.1 só permite rodar um objeto sobre ele próprio (tente usar o mesmo código com os dois cubos com x= -2 e x=2 e veja o resultado).

Para permitir mudar o ponto de vista é necessário agir sobre a posição e orientação da câmara e não sobre a posição e orientação do objeto.

Modifique o código da alínea 4.1 anterior para movimentar agora a câmara sobre uma esfera centrada na origem e com raio 5. Deve permitir atualizar a posição da câmara na esfera de acordo com as variáveis pi e theta. Note que para além da posição (`camera.position.set(x,y,z)`) tem também de definir a orientação da câmara (`camera.lookAt()`). Para calcular as coordenadas cartesianas (x,y,z) a partir das coordenadas esféricas (rho, phi, theta). Pode usar o código seguinte:

```
theta = …
phi = …
camera.position.x = raio * Math.sin(theta) * Math.cos(phi);
camera.position.y = raio * Math.sin(phi);
camera.position.z = raio * Math.cos(theta) * Math.cos(phi);
camera.updateMatrix();
```

Utilize as teclas +/- do teclado (ver a última aula) para permitir fazer “zoom in” e “zoom out”.

4.4 Texto

Utilize o `TextGeometry` do Three.js para colocar em cima dos cubos um texto indicando “cubo1” ou “cubo2”. Utilize a fonte `"helvetiker"` (pode utilizar o ficheiro fornecido na pasta examples/fonts do three.js (`helvetiker_regular.typeface.json`). Para definir o objeto de texto pode usar o código seguinte para criar a geometria de texto para o cubo 1.

```
var textMesh1;
var loader = new THREE.FontLoader();
 loader.load("../../js/helvetiker_regular.typeface.json", function
(font) {
      textGeometry1 = new THREE.TextGeometry("Cube 1", {
          font: font,
          size: 0.22,
          height: 0.05,
      });
      var materialText = new THREE.MeshBasicMaterial({ color: 0x00ff00
});
      textMesh1 = new THREE.Mesh(textGeometry1, materialText);
      textMesh1.position.x = -2.5;
      textMesh1.position.y = 1.5;
});
```

Altere o código usando o exemplo 4.2 para que o texto só apareça ao selecionar um dado cubo, para tal use de forma adequada a propriedade visible do mesh.