Tutorial 6 – *ThreeJS*

Nos tutoriais anteriores, aprendeste a linguagem de baixo nível para programação em WebGL. No entanto, existem bibliotecas que permitem simplificar a programação em WebGL, permitindo assim uma maior facilidade no desenvolvimento de aplicações. Este tutorial foca-se na utilização de uma das mais conhecidas bibliotecas de *WebGL*, a biblioteca *ThreeJS*.

1. Objetivos de aprendizagem

Com este tutorial, vais aprender a configurar o teu ambiente de desenvolvimento de forma a usares a biblioteca *ThreeJS*. Para além disso, neste tutorial, irás aprender a criar uma cena básica constituída por um cubo e a implementar mecanismos de interação com o mesmo através do rato e do teclado.

2. Tutorial

2.1. Ficheiros necessários

1. Cria uma pasta com o nome "*Tutorial 6*".

2. Acede ao seguinte link: https://nodejs.org/en/ para instalar a aplicação "*Node.JS*". Irás encontrar uma página semelhante à página ilustrada na Figura 1.

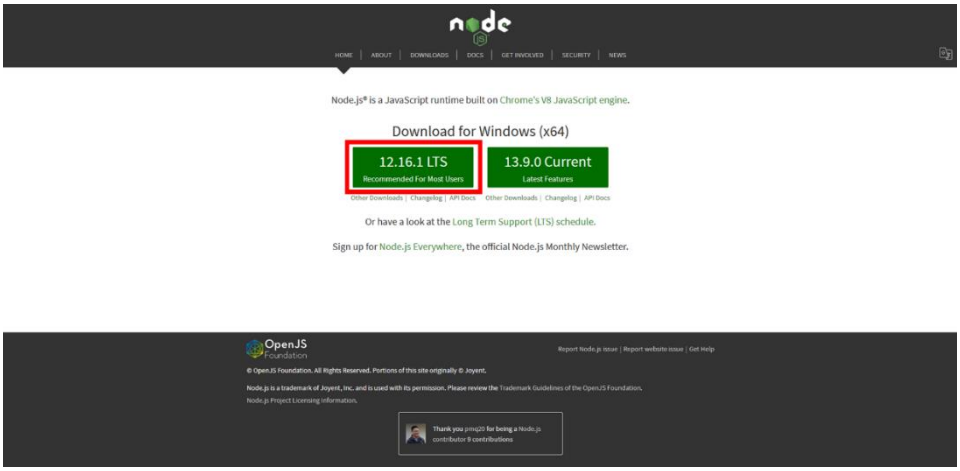

Figura 1 – Página para download da aplicação ThreeJS.

3. Clica no botão que está assinalado a vermelho na Figura 1 para descarregares o instalador da aplicação. Quando acabar de descarregar o instalador, instala a aplicação.

4. Acede ao link https://threejs.org/build/three.js . Será apresentada uma página com o código correspondente à biblioteca *ThreeJS*. Clica com o botão do lado direito do rato em qualquer zona da página. Seleciona a opção "Guardar como" ou "Guardar página

como" e guarda o ficheiro na pasta com o nome "*JavaScript*" dentro da pasta "*Tutorial 6*". O ficheiro deverá ter o nome de "*three.js*".

5. Abre o *VSCode* e, com o *VSCode*, abre a pasta "*Tutorial 6*". De seguida, cria dois ficheiros: o ficheiro "*index.html"* na raiz do projeto e o ficheiro "*app.js*" dentro da pasta "*JavaScript*".

Neste ponto, já tens todos os ficheiros necessários para utilizares a biblioteca *ThreeJS*, mas ainda não tens instalado o *IntelliSense* (que te confere autocomplete/ajudas). Para instalares o *IntelliSense*, terás de te dirigir ao terminal do VSCode (assinalado a vermelho na Figura 2). Caso o terminal não esteja aberto, vai ao menu Terminal → New Terminal ou através do atalho CTRL + SHIFT + Ç.

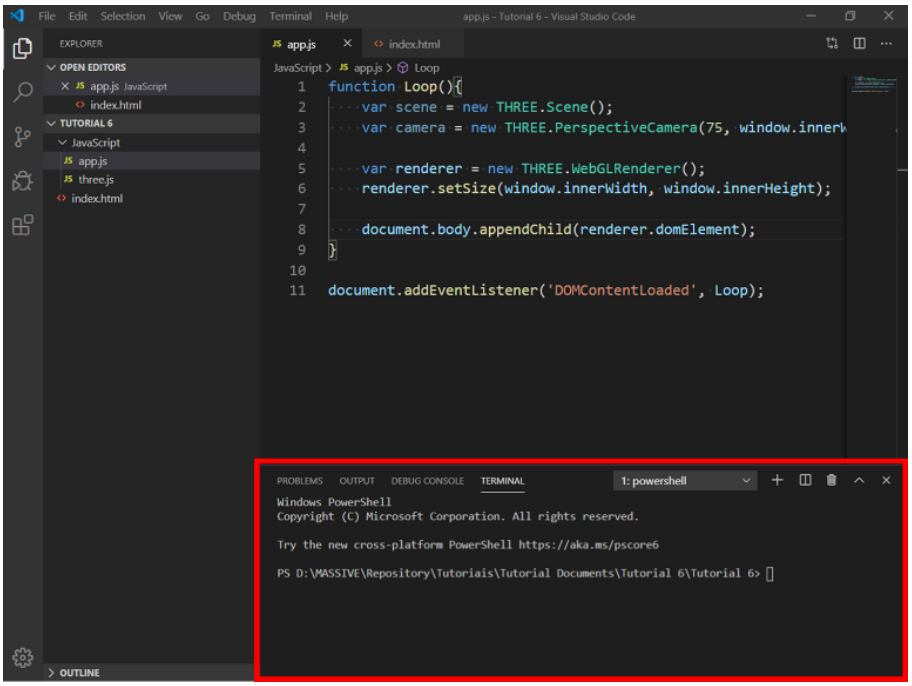

Figura 2 – Terminal do VSCode.

6. De seguida escreve os seguintes comandos[1]:
   - *npm install typings --global* (para instalares um plugin, se usares Mac ou Linux coloca sudo antes do termo npm)
   - *typings init* (para inicializares o plugin)[2]

[1] não copies o que está entre parêntesis "()", isto apenas serve para explicar o que faz cada comando e, no final de cada comando, pressiona *enter*).
[2] Se ocorrer erros na execução deste comando, vai a Ficheiro -> Preferências -> Preferências -> Definições -> Extensões. Faz scroll até ao fundo e carrega em "Editar em settings.json" e coloca a seguinte linha de código:

"terminal.integrated.shellArgs.windows": ["-ExecutionPolicy", "Bypass"]

Depois, reinicia o Visual Studio Code.

- *typings install dt~three --save --global* (instala a biblioteca de ajuda do *ThreeJS*, se usares Mac ou Linux coloca *sudo* antes do termo *npm*). Nota: se der erro, usa o comando *typings install three --save –global*

7. No final da execução destes comandos, deverás adicionar um novo ficheiro à raiz do projeto com o nome "*jsconfig.json*" e copiar o seguinte código para esse novo ficheiro:

```
{
    "typeAcquisition": {
        "include": [
            "three"
        ]
    }
}
```

A partir deste momento já deverás ter *IntelliSense* para te ajudar a escrever código sobre a biblioteca *ThreeJS.*

2.2. Configuração de cena básica em ThreeJS

1. Abre o ficheiro "*index.html*" e transcreve o seguinte código:

```
<!DOCTYPE html>
<html>
    <head>
        <title>
            WebGL - Three.js
        </title>
    </head>
    <body>
        <!-- Adicionar biblioteca three.js -->
        <script src="./JavaScript/three.js"></script>
        <!-- Adicionar o nosso ficheiro javascript -->
        <script src="./JavaScript/app.js"></script>
    </body>
</html>
```

2. Abre o ficheiro "*app.js*" e transcreve o seguinte código:

```
// Indica ao documento HTML que quando acabar de carregar todo o seu conteúdo
// deve chamar a função "Start".
document.addEventListener('DOMContentLoaded', Start);

// Em three.js tudo é baseado em cenas e camâras. Cada cena contém os objetos que a ela pertencem.
// Podem existir diferentes camâras mas apenas uma é rederizada.
// As linhas de código abaixo criar uma cena, uma camâra e um render em WebGL.
// Este último é o que vai renderizar a imagem tendo em conta a camâma e a cena.
var cena = new THREE.Scene();
var camara = new THREE.PerspectiveCamera(75, window.innerWidth/window.innerHeight, 0.1, 1000);
var renderer = new THREE.WebGLRenderer();

// O código abaixo indica ao render qual o tamanho da janela de visualização
renderer.setSize(window.innerWidth -15, window.innerHeight-15);

// O código abaixo adiciona o render ao body do documento html para que este possa ser visto.
document.body.appendChild(renderer.domElement);

// Em THREE.JS existem diferentes primitivas entre as quais: Box (cubo), plano (2D), círculo (2D),
// esfera, cone, cilindro,tetraedro, dodecaedro, icosaedro, octaedro, poliedro, ring (2D),
// geometria para texto e torus. Existem mais algumas, se quiseres saber mais vai a
// "https://threejsfundamentals.org/threejs/lessons/threejs-primitives.html".
// Para criarmos um objeto precisamos sempre de uma geometria e um material, o primeiro é
// responsável por definir a geometria (ou vértices de cada ponto), e o segundo é responsável
// por dizer qual o material que o objeto irá usar.
// Para criar um cubo é necessário criar a geometria para isso utilizamos o código abaixo indicando
// qual o comprimento, altura e profundidade, respetivamente.
var geometria = new THREE.BoxGeometry(1,1,1);
// É necessário também criar o material, para este caso vamos utilizar um material básico e
// dentro desse material básico (que representa uma cor) mudamos o parâmetro color para ser
// vermelho em hexadécimal.
var material = new THREE.MeshBasicMaterial({color: 0xff0000});

// No final, quando já tens a geometria e o material, é necessário criares uma mesh
// com os dados da geometria e do material. A Mesh é o componente necessário para
// poderes fazer as diferentes transformações ao objeto.
var cubo = new THREE.Mesh(geometria, material);


// Função chamada quando a página HTML acabar de carregar e é responsável por configurar
// a cena para a primeira renderização.
function Start(){
    // O código abaixo adiciona o cubo que criamos anteriormente à cena.
    cena.add(cubo);

    // Coloca a camâra a 6 unidades no eixo do Z apartir do centro do mundo.
    camara.position.z = 6;

    // Como já sabes, esta linha de código é reposável por dizer ao browser que
    // pretendemos criar uma animação e que deve chamar a função passada no parâmetro
    // da função.
    requestAnimationFrame(update);
}

// Função chamada a cada frame para poder-mos criar animações. Caso contrário
// apenas veriamos uma simples imagem sem haver mudanças.
function update(){
    // A cada frame mudamos a rotação do cubo no eixo do X.
    cubo.rotation.x += 0.1;

    // Renderizamos a cena tendo em conta qual a cena que queremos visualizar e
    // a camâra que pretendemos.
    renderer.render( cena, camara);

    // Como já sabes, esta linha de código é reposável por dizer ao browser que
    // pretendemos criar uma animação e que deve chamar a função passada no parâmetro
    // da função.
    requestAnimationFrame(update);
}

```

3. Depois de copiares todo o código da imagem acima, poderás abrir a página *HTML* com recurso ao *Live server* e verificar que tens uma cena básica com um cubo vermelho a rodar no eixo do *x*.

2.3. Interação em ThreeJS

1. Para rodar o cubo dependendo da posição do rato no ecrã e movimentar a câmara utilizando as teclas *W*,*A*,*S* e *D* do teclado, utiliza o seguinte código para o ficheiro "apps.js", na zona onde são declaradas as variáveis:

```
39
40  // Variável que vai guardar que rotação aplicar ao cubo.
41  var cuboCoordRotation;
42
43  // variável que irá guardar para que direções a camara se irá movimentar.
44  var camaraAndar = { x:0, y:0, z:0};
45
46  // Velocidade de movimentação que a camara irá andar.
47  var velocidadeAndar = 0.05;
48
49
50  // Este código adiciona um evento que é deplotado sempre que o rato se mexer
51  document.addEventListener('mousemove', ev =>{
52      // A posição do rato encontra-se de 0 até ao tamanho do ecrã em pixeis. É então
53      // necessário converte-lo para a escala de -1 a 1. Para isso utilizamos o código
54      // a baixo.
55      var x = (ev.clientX - 0)/ (window.innerWidth - 0) * (1 - (-1)) + -1;
56      var y = (ev.clientY - 0) / (window.innerHeight - 0) * (1 - (-1)) + -1;
57
58      // Adicionamos a rotação que devemos aplicar na variável cuboCoordRotation.
59      cuboCoordRotation = {
60          x:x,
61          y:y
62      }
63  } );
64
```

```
65
66  // Adicionamos um evento que é desplotado sempre que uma tecla for mantida pressionada.
67  document.addEventListener('keydown', ev=>{
68      // Inicializa a variável de controlo
69      var coords = {
70          x:0,
71          y:0,
72          z:0
73      };
74      // Verifica se a tecla W foi premida e coloca a variável Z do coords para o valor correto
75      if(ev.keyCode == 87)
76          coords.z -= velocidadeAndar;
77      // Verifica se a tecla S foi premida e coloca a variável Z do coords para o valor correto
78      if(ev.keyCode == 83)
79          coords.z += velocidadeAndar;
80      // Verifica se a tecla A foi premida e coloca a variável Z do coords para o valor correto
81      if(ev.keyCode == 65)
82          coords.x -= velocidadeAndar;
83      // Verifica se a tecla F foi premida e coloca a variável Z do coords para o valor correto
84      if(ev.keyCode == 68)
85          coords.x += velocidadeAndar;
86
87      // Aplica a variável coords à variável camaraAndar;
88      camaraAndar = coords;
89  });
90
```

```
// Adicionamos um evento que é desplotado sempre que uma tecla deixar de ser premida.
document.addEventListener('keyup', ev=>{
    // Inicializa a variável de controlo
    var coords = {
        x:0,
        y:0,
        z:0
    };

    // Verifica se a tecla W foi levantada
    if(ev.keyCode == 87)
        coords.z += velocidadeAndar;
    // Verifica se a tecla S foi levantada
    if(ev.keyCode == 83)
        coords.z -= velocidadeAndar;
    // Verifica se a tecla A foi levantada
    if(ev.keyCode == 65)
        coords.x += velocidadeAndar;
    // Verifica se a tecla D foi levantada
    if(ev.keyCode == 68)
        coords.x -= velocidadeAndar;

    // Aplica a variável coords à variável camaraAndar;
    camaraAndar = coords;
});
```

2. Na função update(), substitui a primeira linha de código pelo seguinte:

```
function update(){

    // A cada frame mudamos a rotação do cubo no eixo tendo em conta a posição do rato.
    if(cuboCoordRotation != null)
    {
        cubo.rotation.x += cuboCoordRotation.y * 0.1;
        cubo.rotation.y += cuboCoordRotation.x * 0.1;
    }

    // A cada frame mudamos a posição da camara tendo em conta as teclas premidas.
    if(camaraAndar != null)
    {
        camara.position.x += camaraAndar.x;
        camara.position.z += camaraAndar.z;
    }

    // Reiniciar a variável
    camaraAndar = {x:0, y:0, z:0};

    // Renderizamos a cena tendo em conta qual a cena que queremos visualizar e
    // a camara que pretendemos.
    renderer.render( cena, camara);

```

Agora, se abrires a página *HTML* novamente, irás verificar que o cubo irá rodar tendo em conta o movimento do rato. Para além disso, se premires as teclas *W*, *A*, *S* ou *D*, irás ver que a câmara se move pela cena.

3. Desafios

Desafio 1. Faz com que, cada vez que pressionas a barra de espaço, seja renderizado um cubo na cena numa posição aleatória e com uma cor aleatória[3][4].

[3] Acede ao link http://gcctech.org/csc/javascript/javascript_keycodes.htm para consultares os códigos das teclas
[4] Podes defenir as cores aleatoriamente através da expressão *setHex(Math.random() * 0xffffff)*

Lembra-te de que o *ThreeJS* permite a geração de números aleatórios através da função *THREE.MATH.randINT(min,max)*[5]

Desafio 2. Com recurso às primitivas disponíveis no *ThreeJS*, constrói um boneco de neve 3D tal como ilustrado na Figura 3[6].

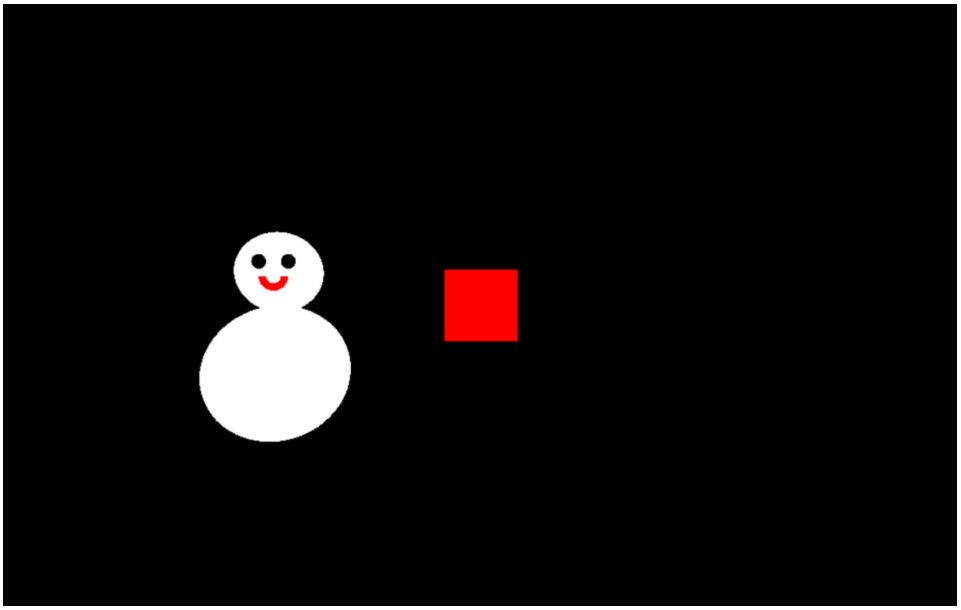

Figura 3 - Captura de ecrã ilustrativas do resultado esperado do Desafio 2.

ENTREGA

O trabalho deve ser submetido no MOODLE até dia **30/04/2021.**

A submissão é individual e deve conter todos os ficheiros necessários à correta execução da aplicação: uma pasta contendo o ficheiro .html e a pasta JavaScript com os respetivos ficheiros.

[5] Acede a https://threejs.org/docs/#api/en/math/MathUtils.randInt para mais informações acerca da função THREE.MATH.randINT(min,max)
[6] A lista de primitivas pode ser consultada no link https://threejsfundamentals.org/threejs/lessons/threejs-primitives.html