

”

E-fólio A | Folha de resolução para E-fólio

UNIDADE CURRICULAR: Computação Gráfica

CÓDIGO: 21020

DOCENTE: António Araújo e Pedro Pestana

A preencher pelo estudante

NOME: Luís Carlos Crispim Pereira

N.º DE ESTUDANTE: 2300163

CURSO: LEI – Licenciatura em Engenharia Informática

DATA DE ENTREGA: 23/11/25

TRABALHO / RESOLUÇÃO:

Pergunta 1:

1. F
2. V
3. V
4. V
5. F
6. V

Pergunta 2:

A implementação foi desenvolvida em conformidade com o enunciado do e-fólio A, utilizando exclusivamente HTML, CSS e JavaScript com a biblioteca Three.js importada via CDN, sem qualquer dependência adicional. A solução foi estruturada para manter clareza, modularidade e total correspondência com os requisitos funcionais e visuais definidos.

O primeiro bloco de código cria o ambiente gráfico, iniciando com o elemento HTML que serve de área de renderização. Definiu-se um viewport quadrado através de CSS, uma vez que o enunciado exige uma janela proporcional de 1:1 para a visualização da cena. O body foi configurado com um fundo preto e mecanismos de centragem para garantir que a área de desenho permanece visualmente equilibrada em qualquer resolução do ecrã.

Seguiu-se a criação da cena (`THREE.Scene()`), do renderer WebGL e das fontes de iluminação. A luz ambiente foi incluída para garantir uma iluminação uniforme mínima, enquanto a luz direcional foi posicionada de forma a evidenciar volume e profundidade nos objetos. Esta combinação permite realçar o efeito tridimensional das esferas e do cubo, respeitando a estética observada nos exemplos fornecidos pelo docente.

No que diz respeito à modelação dos objetos, adotou-se um raio fixo para a esfera inferior, que funciona como elemento de referência de toda a cena. O cubo inscrito foi construído com lado igual ao diâmetro dessa esfera, garantindo assim a condição geométrica adequada, um cubo inscrito numa esfera apresenta exatamente essa relação dimensional. O material do cubo foi definido como semitransparente, tal como solicitado, o que permite observar a relação espacial entre os objetos.

A esfera superior utiliza a mesma geometria base, mas o seu “raio” é simulado através de alterações na escala do mesh. A fórmula utilizada ($0.5 + 0.5\cos(\omega t)$) como recomendado pelo docente, produz uma oscilação suave e periódica, com ciclo de quatro segundos, também exigido no enunciado. A sua posição vertical é continuamente ajustada para garantir contacto constante com a esfera inferior, assegurando assim a coerência física da animação.

A câmara foi posicionada manualmente após testes visuais, de modo a apresentar uma perspetiva clara e equilibrada que permite observar simultaneamente as duas esferas e o cubo inscrito. O `lookAt()` aponta para o ponto de contacto entre as duas esferas, reforçando a legibilidade visual da animação.

Em relação ao render, definiu-se uma função de redimensionamento que mantém sempre a proporção quadrada do viewport, ajustando o renderer e a câmara sempre que necessário. Isto garante que a animação permanece correta e não sofre distorções mesmo no caso de variações do tamanho da janela do utilizador.

Por fim, a função de animação utiliza o relógio interno do Three.js para manter sincronização temporal rigorosa. O cubo executa uma rotação completa a cada quatro segundos, em perfeita correspondência com a oscilação da esfera superior, proporcionando uma cena coerente, estável e fluida.

Esta abordagem garante simplicidade, legibilidade e fidelidade total ao enunciado, preservando ao mesmo tempo uma organização clara do código para facilitar futuras extensões ou manutenção.

Código:

```
<!DOCTYPE html>
<html lang="pt">

<head>
  <meta charset="UTF-8">
  <title>eFólio A - Computação Gráfica | Luís Pereira 2300163</title>

  <style>
    body {
      /* Fundo e centragem da área de renderização */
      margin: 0;
      background: #000;
      display: flex;
      justify-content: center;
      align-items: center;
      height: 90vh;
    }

    /* Viewport quadrado conforme enunciado */
    #zona {
      width: 80vmin;
      height: 80vmin;
    }
  </style>
</head>

<body>
<div id="zona"></div>

<script type="module">

  /* Importação da biblioteca THREE.js conforme enunciado */
  import * as THREE from 'https://unpkg.com/three@0.132.0/build/three.module.js';

  /* Criação do renderer e ligação ao DOM (Document Object Model) */
  // Obtém a div onde irá ser renderizada a cena
  const quadro = document.getElementById("zona");

  // Cria o motor WebGL com suavização
  const motor = new THREE.WebGLRenderer({ antialias: true });
```



```

// Insere o canvas gerado pelo motor na página
quadro.appendChild(motor.domElement);

/* Criação da cena e definição do fundo */
// Cria a cena onde serão inseridos os objetos
const cena = new THREE.Scene();

// Define o fundo a preto conforme o enunciado
cena.background = new THREE.Color(0x000000);

/* Luz ambiente para iluminação geral */
// Luz suave que ilumina todos os objetos igualmente
const luzAmb = new THREE.AmbientLight(0xffffff, 0.4);
cena.add(luzAmb);

/* Luz direcional para dar volume à cena */
// Define direção e intensidade da luz principal
const luzTopo = new THREE.DirectionalLight(0xffffff, 1);
luzTopo.position.set(10, 12, 6);
cena.add(luzTopo);

/* Construção das esferas e do cubo */
// Raio fixo da esfera inferior
const raioInferior = 2;

// Geometria base utilizada para ambas as esferas
const esferaGeo = new THREE.SphereGeometry(raioInferior, 32, 32);

// Material vermelho comum às duas esferas
const matVerm = new THREE.MeshStandardMaterial({ color: 0xff3333 });

// Esfera inferior fixa
const esferaA = new THREE.Mesh(esferaGeo, matVerm);
esferaA.position.y = 0;
cena.add(esferaA);

// Esfera superior cujo raio será animado
const esferaB = new THREE.Mesh(esferaGeo.clone(), matVerm);
cena.add(esferaB);

// Lado do cubo igual ao diâmetro da esfera inferior
const lado = raioInferior * 2;

// Geometria cúbica
const cuboGeo = new THREE.BoxGeometry(lado, lado, lado);

// Material verde semitransparente para o cubo
const matVerde = new THREE.MeshStandardMaterial({
    color: 0x00ff00,
    transparent: true,
    opacity: 0.5
});

// Cubo inscrito na esfera inferior
const cubo = new THREE.Mesh(cuboGeo, matVerde);
cena.add(cubo);

/* Configuração da câmara */
// Câmara com campo de visão de 40° e proporção fixa
const camara = new THREE.PerspectiveCamera(40, 1, 0.1, 200);

// Posição escolhida para enquadrar as duas esferas e o cubo
camara.position.set(15, 4, 10);

// A câmara aponta para o ponto onde as esferas se tocam
camara.lookAt(0, raioInferior, 0);

/* Função para manter sempre um viewport quadrado */
function fixRender() {
    // Obtém o lado mínimo para construir o quadrado
    const s = Math.min(quadro.clientWidth, quadro.clientHeight);

    // Aplica o tamanho ao renderer
    motor.setSize(s, s);
}

```



```

        // Mantém a proporção da câmara em 1:1
        camara.aspect = 1;
        camara.updateProjectionMatrix();
    }

    // Atualiza o viewport ao redimensionar a janela
    window.addEventListener("resize", fixRender);

    // Aplica o tamanho inicial
    fixRender();

    /* Parâmetros da animação (4 segundos por ciclo) */
    // Relógio interno para medir o tempo decorrido
    const relógio = new THREE.Clock();

    // Duração completa do ciclo
    const time = 4;

    // Velocidade angular que garante um ciclo de 4 segundos
    const omega = (2 * Math.PI) / time;

    /* Função principal de animação */
    function ciclo() {

        // Tempo decorrido desde o início
        const t = relógio.getElapsedTime();

        // Raio variável da esfera superior (oscilação suave)
        const r2 = raioInferior * (0.5 + 0.5 * Math.cos(omega * t));

        // Escala calculada para ajustar a esfera superior
        const escala = r2 / raioInferior;
        esferaB.scale.set(escala, escala, escala);

        // Mantém a esfera superior sempre em contacto com a inferior
        esferaB.position.y = raioInferior + r2;

        // O cubo roda uma volta completa em 4 segundos
        cubo.rotation.y = omega * t;

        // Renderização da cena
        motor.render(cena, camara);

        // Pedido do próximo frame da animação
        requestAnimationFrame(ciclo);
    }

    // Inicia a animação
    ciclo();
</script>
</body>
</html>

```