

**UNIDADE CURRICULAR:** Computação Gráfica

**CÓDIGO:** 21020

**DOCENTE:** Pedro Pestana e António Araújo

**A preencher pelo estudante**

**NOME:** Mário Pedro Capela Rodrigues Carvalho

**N.º DE ESTUDANTE:** 2000563

**CURSO:** Licenciatura em Engenharia Informática

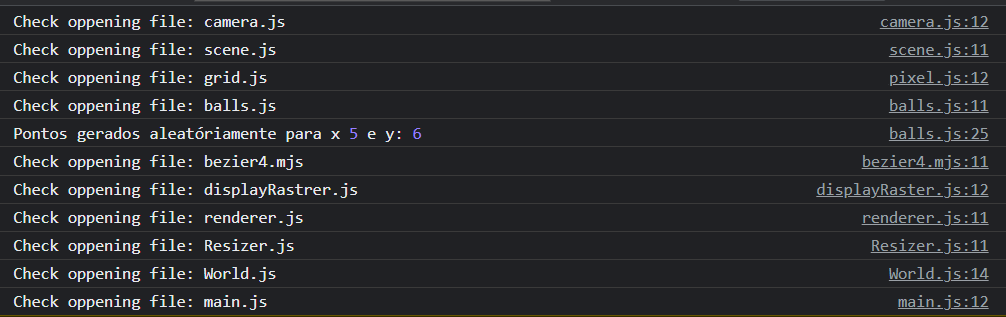
**DATA DE ENTREGA:**

**TRABALHO :**

Este relatório tem como objetivo de explicar o trabalho realizado em paralelo, até ter sido alcançado o objetivo final do e-fólio em questão. Este, demonstra assim, o que fora pensado em cada etapa, e como se foi ponderando cada parte da sua construção, para ser realizado com sucesso. Assim, antes de se iniciar qualquer parte de código, ou reproduzir as condições necessárias, teve-se em atenção em corrigir o problema de fluidez não detetado anteriormente, com uma nova implementação do “orbitControls”, e de seguida, manteve-se a arquitetura do e-fólio anterior, eliminando apenas os ficheiros desnecessários, e adaptando/ restruturando as partes necessárias para responder ao pedido no enunciado: organização das pastas, nomes, recursos aos “imports” necessários – CDN, three.js, ....

Toda a restante estrutura foi sendo realizada por partes nos respetivos ficheiros, com importação de uma estrutura idêntica e bem dividida segundo a documentação de “threejs.org”, com os ficheiros pedidos (index.html, bezier4.mjs, ReadMe.pdf) na sua base, e os restantes, subdivididos dentro de “./src”, onde constam três grupos : “./styles” para o ficheiro correspondente a “.css”, “./World”, que tem “./components” e “systems”, e “main.js”, que importa o ficheiro “World.js”. Esta subdivisão, tal como indicada na documentação de “three.js”, permite de forma mais segura, limitar o acesso às propriedades de cada objeto a partir de uma fonte externa, e simultaneamente, facilitar a sua importação para outro projeto de forma idêntica, sem que esta sofra modificações ou altere o projeto seguinte.

Deste modo, em “./systems”, correm os ficheiros “renderer.js”, e “Resizer.js”. Em “./components”, correm os ficheiros “camera.js”, “scene.js”, “./pixel.js”, “./displayRaster.js”.

Além de se ter atribuído particular atenção à Parte I, para se garantir que o cálculo da função de bezier quártica, permitiria calcular com certeza a curva, pelos pontos *Ci*, realizaram-se vários cálculos posteriormente aquando da sua implementação da função de “bezier4.mjs”, no respetivo ficheiro. Uma vez que a Parte I – Implementação do algoritmo da curva de Bézier quártica, foi concluída, iniciou-se a implementação da Parte II – Interface Gráfico. Para melhor compreensão de qual o ficheiro que estaria a ser usado, e tornar mais visual a ordem em que cada ficheiro se processava, foi introduzido um “console.log” indicativo com o nome de cada ficheiro.

De forma a continuar com a estrutura aconselhada por ThreeJs.org, e dado que seria necessário manter o tabuleiro do é-folio precedente, foi acrescentado à pasta “\components”, o ficheiro “balls.js”. Consiste em conceber os parâmetros de cada bola, em termos de nome, cor e posição, para que, só na sua importação para o ficheiro “displayRaster.js”, é que estas sejam criadas, segundo o seu objeto. Segundo o enunciado, interpretou-se que as bolas teriam de estar num primeiro momento, no plano, mas que, a cada início a posição das suas coordenadas (x,y), seriam aleatórias dentro do tabuleiro. Para facilitar a sua implementação, criou-se neste momento, uma coordenada random para x e outra para y, de forma a obter números exatos (int). Como se tratam de cinco bolas diferentes, e existem apenas 4 combinações de coordenadas diferentes possíveis (para dois números, em positivo e/ou negativo), recorreu-se (por tentativas e erros) a subtrações e /ou somas com um ou os dois números, o próprio numero obtido, e/ou alteração de sinais, para que se conseguisse colocar 4 das 5 bolas em posições diferentes entre si, e que tendencialmente não ficassem no ponto 0 nem no mesmo ponto, desde que, ficassem sempre dentro do tabuleiro. A esta etapa dedicou-se algum tempo, pois, o fenómeno aleatório não propriamente, matemático.

Recorreu-se a “camara.js”, para obrigar que, ao se iniciar a página, se observe como posição central do tabuleiro, a posição (0,0,0) onde se cruzam os 3 eixos, a incidir no ponto de cruzamento dos quatro pixels centrais, com recurso a “camera.lookAt()”. Como neste caso os pontos centrais entre eixos e coordenadas centrais, não se encontravam alinhados no mesmo ponto do tabuleiro (e que este, também teria de estar centrado), recorreu-se para isso, à mudança de pontos iniciais de cada pixel do tabuleiro, para um deslocamento de 0.5 em x e em y. Tendo para isso, que se manter o ficheiro “pixel.js” como este se encontrava para a construção do tabuleiro.

A restante implementação, foi realizada diretamente no “displayRaster.js”, que permitiu acrescentar e manipular a restante construção: bolas, eventos para a interação com o utilizador, e os recursos de visualização (linhas perpendiculares, tubo, ..).

Começou-se por adicionar um novo tamanho de tabuleiro, e um eixo z aos restantes. Para este último, assumiu-se que ficaria com uma cor diferente das restantes. Por mais que tenha semelhanças com a função original de “axesHelper()” preferiu-se adicionar mais uma construção, em vez de utilizar algo próprio da biblioteca de “Threejs”. Desta forma, conseguiu-se readaptar o tamanho destas linhas à nova dimensão do tabuleiro.

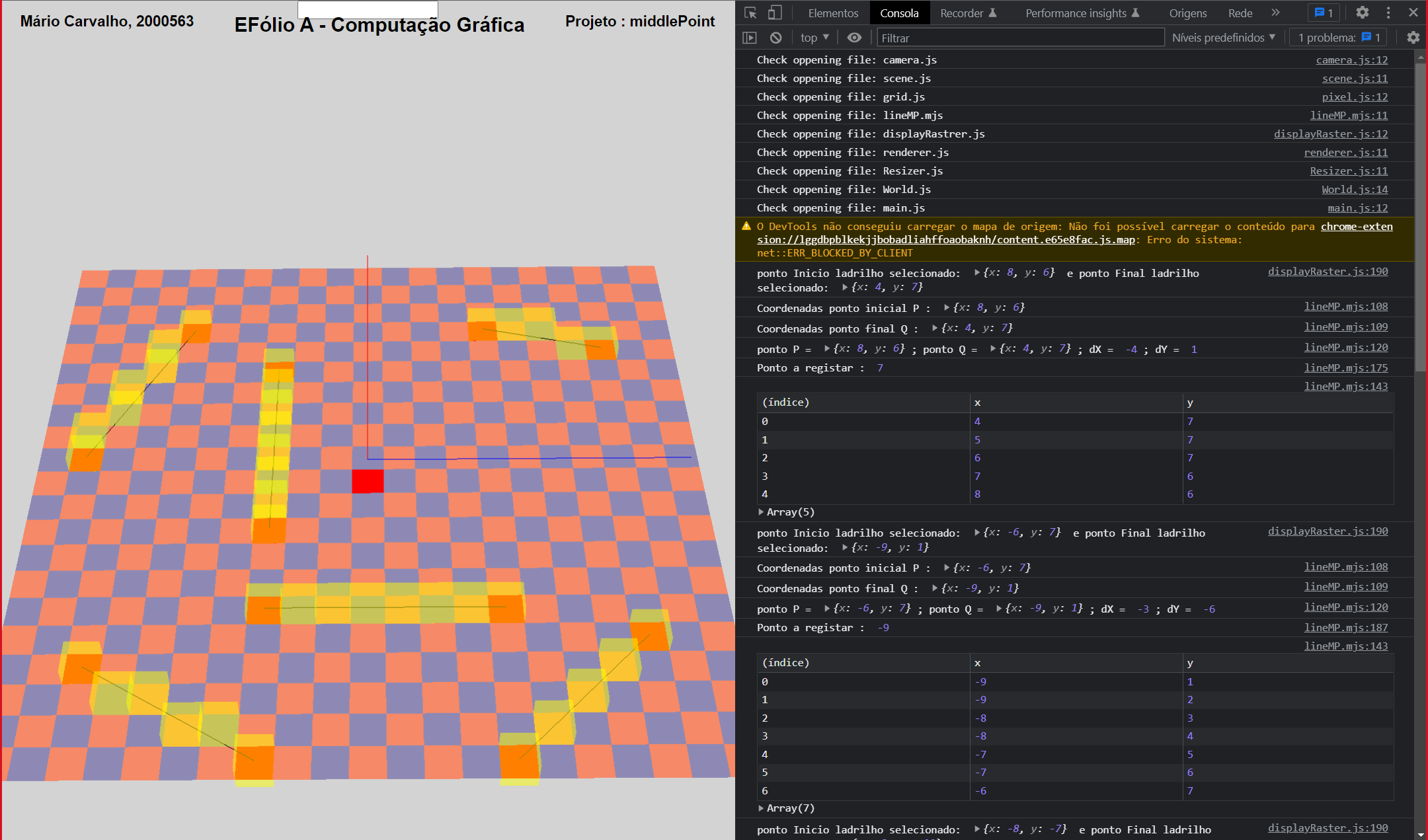
Criaram-se novas funções que permitiriam o surgimento da estrutura de cada bola (“createSphere()”), e a sua construção (“createBalls()”) para o seu surgimento no tabuleiro, depois de adicionados com recurso a “scene.add()”.

Seleção de boldas,

coordenadas updates,

events e

linhas



A “index.html” foi adicionado um explicativo instrutivo e sobre os controlos para a sua utilização, como também uma zona onde se fixariam as coordenadas das bolas selecionadas. Este ponto, teve a sua dificuldade, mas, após consulta sobre o modo de implementar algo dinâmico entre a posição do rato e front-end, com recurso a “document.creatElement(‘div’)”, a sua implementação ficou acessível, trantando-se apenas de criar uma nova manipulação entre os diferentes elementos, para quando existe bola selecionada e para quando não existisse bola selecionada.

Para efeitos de clareza sobre o modo de interagir com a interface, foi acrescentado um texto, sobre o modo de funcionamento do projeto e interação com o tabuleiro e bolas.

Para fazer funcionar o projeto, utilizou-se como servidor o aconselhado no fórum da disciplina (‘Web Server for Chrome’).

O e-fólio encontra-se em modo partilha no github, a partir do dia seguinte à data limite de entrega, em [repositório do estudante](mailto:https://github.com/MP-C/UaB.ComputacaoGraficaEfolioA?subject=repositório%20do%20estudante), para facilitar acesso ao trabalho realizado mesmo em caso de dificuldades de abertura deste documento partilhado em modo “Carvalho2000563.zip”.