

”

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



UNIDADE CURRICULAR: Computação Gráfica

CÓDIGO: 21020

NOME: David Miguel Raposo Ferreira

N.º DE ESTUDANTE: 2102841

CURSO: LEI

DATA DE ENTREGA: 06/11/2023

TRABALHO / RESOLUÇÃO:

Para resolver este efólio comecei pela parte 1 que consiste no desenvolvimento do algoritmo do ponto médio. Para desenvolver este algoritmo comecei por fazer alguma pesquisa sobre o que é, tanto nos links fornecidos pelo professor como no youtube e outros websites. Utilizei console.log para verificar que realmente apresentava os pontos do exemplo do efólio.

Para resolver a parte 2 comecei por criar o ficheiro index.html que apenas que carrega o script main.mjs. Para o script fui lendo o efólio e apontando em partes as *features* necessárias para sua resolução. Comecei por importar todas as dependências necessárias e ver algumas noções básicas de three.js.

Criei todos os componentes necessários para criar uma cena, incluindo uma cena, uma camera, um renderer e um raycaster. Depois o meu primeiro plano foi criar uma grelha composta por Mesh de planos. Criei uma função para os desenhar e ao mesmo tempo desenhar também os eixos das abcissas e das ordenadas.

Depois de conseguir desenhar a grelha na cena, adicionei os OrbitControls, criando uma nova variável (sem necessidade de dar assign) e comecei a “brincar” com o raycaster e o console.log para ver o que aparecia. Criei uma função para o eventListener “pointermove” para imprimir as coordenadas do plano sob o qual o rato se encontra. A primeira vez que criei os planos da grelha, quando lhes atribui um nome, adicionei ao nome as coordenadas para mais tarde cortar a string para obter as coordenadas sem ter de aceder ao objeto, mas rapidamente percebi que era mais fácil aceder às coordenadas diretamente. Alterei o método para apenas imprimir as coordenadas uma vez enquanto o rato não sair do plano, isto é, quando o rato se mexe, o evento “pointermove” é chamado várias vezes, pelo que iria imprimir várias vezes as coordenadas do plano em que se encontra, então criei algumas condições para isso não ocorrer.

Depois comecei a mexer nas cores e a ver como é que os objetos são afetados e peguei numa lista de cores para cada tipo de interação que consegui identificar (cores dos planos, cores das caixas das linhas, etc).

De seguida comecei a tratar os inputs do teclado, sendo que quando carrega no X seleciona um plano e ao selecionar dois planos, chama o algoritmo do ponto médio entre esses dois planos. Para criar essa função crio a linha preta entre os dois e torno visíveis as caixas com $\frac{1}{4}$ de altura em relação ao lado dos planos. Quando se carrega no C, desloco a posição da câmara até à posição inicial. Considerei usar TWEEN para tratar o movimento, mas como o enunciado não especifica se são permitidas dependências externas, decidi usar apenas as propriedades da câmara. Tive alguns problemas no browser quando tentei impedir o utilizador de ter acesso aos controlos enquanto a câmara se move, mas acabei por encontrar a solução no evento que chama a função que mexe a câmara.

Em relação ao vídeo de exemplo, utilizei o software OBS para gravar e Davinci Resolve para editar. No video apresento todas as funcionalidades, inclusive o que aparece na consola do browser.

Toda a bibliografia ou webgrafia que usei para todo o desenvolvimento do efólio encontra-se identificada nos ficheiros .mjs com os detalhes sobre a sua utilização. Decidi deixar lá em vez de neste relatório, sendo que apenas comecei a fazer este relatório quando acabei o código e já estava a pôr no código como comentário enquanto as usava.