

”

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



UNIDADE CURRICULAR: Computação Gráfica

CÓDIGO: 21020

DOCENTE: António Araújo e Pedro Pestana

A preencher pelo estudante

NOME: Hélio Emanuel Soares de Sousa

N.º DE ESTUDANTE: 2000027

CURSO: Licenciatura em Engenharia Informática

DATA DE ENTREGA: 04 de janeiro de 2023

TRABALHO / RESOLUÇÃO:

PARTE I- Implementação da curva de Bézier de grau 4 (2 valores):

No módulo bezier4.mjs foi implementada a curva de Bézier de grau 4 tendo por base a equação quádrlica definida no enunciado:

$$f(t) = (1-t)^4 C_0 + 4(1-t)^3 t C_1 + 6(1-t)^2 t^2 C_2 + 4(1-t) t^3 C_3 + t^4 C_4, 0 \leq t \leq 1$$

A abordagem que segui foi:

- Primeiro definir funções para o cálculo individual de cada um dos termos
- Segundo definir funções para a soma dos termos
- Terceiro definir uma função que calcula a posição, em cada de um dos três eixos, da curva de Bézier dada por quatro pontos num dado instante t.

O retorno da função é um ponto no espaço tridimensional dado guardado num objeto Vector3.

PARTE II- Interface Gráfico (2 valores)

O interface gráfico apresentado aproveita grande parte do trabalho desenvolvido no efólio A, bem como a estruturação do código.

Começando pelo index.html, desta vez optei por colocar a informação estritamente necessária, dado que as instruções estão disponíveis no efólio, é acrescentado um style para um novo DIV designado #coord que servirá para mostrar as coordenadas da bola selecionada, aparece e desaparece consoante se seleciona ou desseleciona. O index.html, chama o ficheiro main.js que contém todo o código da interface gráfica.

O ficheiro main.js contém, aproximadamente, 350 linhas de código, contabilizando espaços e comentários, pelo que optei por manter em apenas um ficheiro todo o código, facilita a leitura e compreensão do mesmo. Será relativamente fácil num futuro próximo, subdividir o ficheiro em múltiplos módulos, mas para o efeito de um efólio torna-se moroso a depuração e procura pelas funções e as ligações entre as mesmas. Desta forma, acredito que, a leitura do código será bastante acessível e perceptível para quem não tenha participado no processo de criação do mesmo.

Quanto ao conteúdo do ficheiro:

No início é feita a importação do THREE versão 0.132.2, o OrbitControls e o modulo bezier4.mjs.

De seguida, defino as variáveis globais das quais se destacam as novas em relação ao efólioA:

- Array balls para guardar as esferas criadas (pontos para a curva de Bezier).
- Array ballsLines para guardar as retas finas que unem as esferas ao tabuleiro
- mouseIntersects para guardar o ponto de interseção do raycaster com o tabuleiro para mostrar
- verticalPressed para saber quando existe um movimento vertical e o número da bola selecionada.
- selBall para guardar informações sobre a bola selecionada, tem um booleano para seleção.

Os parâmetros da cena são iguais ao efólioA.

A função start, tem a criação do tabuleiro com o displayRaster, a criação dos três eixos com createAxisHelper, a criação das esferas(bolas) e, por fim, a criação das linhas das esferas.

A animação recursiva é igual ao realizado no efólioA.

De seguida temos os eventos, que tal como o efólio A são apenas de dois tipos: movimento do rato e pressionar de uma tecla.

O movimento do rato é obtido em relação à janela, e se existir uma bola selecionada e não existir movimentos verticais a bola é movimentada com o movimento do rato para a posição pretendida pelo utilizador, ao mesmo tempo o mouseintercepts é atualizado com coordenadas para chamar a função updateCoordinates para que as mesmas possam ser mostradas ao utilizados no ecrã.

O pressionar das teclas é ativado para as teclas 1,2,3,4,5 quando nenhuma bola está selecionada, a tecla pressionada é convertida em dígito inteiro e usada para selecionar a esfera que se pretende movimentar. Caso uma bola esteja selecionada, é possível subir ou descer pressionando as teclas W e S e ao mesmo tempo altera o booleano verticalPressed para true para evitar que os movimentos do rato alterem a posição XY enquanto sobe ou desce a bola. Para guardar a posição final, pressiona-se no espaço que chama a função para deselegionar a bola e passa a false o verticalPressed. As coordenadas são diretamente alteradas no evento com incrementos de 0,1.

A tecla X, cria o tubo, pode sempre ser chamada permitindo fazer diferentes curvas à medida que se move a bola. Por fim, o backspace ser chamado sem restrições para reiniciar a cena.

Após os eventos, são definidas as funções auxiliares:

createAxisHelper – cria os 3 eixos, retirada diretamente da página oficial do Three.js.

createBalls – são criadas 5 esferas a partir da função createSphere e adicionadas à cena.

ballSelected – a base da função é o número que recebe como argumento que permite realizar as operações sobre a esfera selecionada como: transparência, mudar e mostrar as coordenadas.

ballDeselected – torna novamente transparente a esfera, faz a atualização das linhas e remove o DIV do ecrã, desta forma temos o aparecer e desaparecer das coordenadas ao selecionar / desselecionar.

showCoordenates – cria um div com as informações da posição da bola selecionada.

updateCoordenates – atualiza as coordenadas enquanto se move o rato.

removeCoordenates – é a função responsável por apagar o div das coordenadas da bola selecionada.

displayraster – é a mesma função do efolioA.

createPixel – é a mesma função do efolioA, com a ligeira alteração de fazer +0,5 aos eixos X e Y para responder ao enunciado e o ponto (0,0,0) ficar no meio de quatro quadriculas (pixéis).

createSphere – esta função aproveita os métodos e propriedades do objeto Object3D do Three.js para ao criar uma esfera guardar, a cor da esfera, o seu nome e o seu id. A posição inicial é aleatória, para isso utilizei um random com valor máximo igual ao tamanho máximo do tamanho do tabuleiro tanto, em negativo como em positivo, e a posição do eixo do z é definida como zero, como solicitado pelo enunciado.

createBezier4 – de todas a mais difícil de colocar a funcionar, esta função utiliza uma extensão da classe THREE.curve, depois utiliza uma função interna getpoint que ao receber uma variável varia entre 0 e a sua escala, logo getpoint(t) faz t entre 0 e 1. E é aqui, que se introduz o modulo bezier4.mjs para em cada instante t calcular um ponto novo.

Depois, define-se um caminho em path, que irá receber o retorno da classe bezierCurvePath, ou seja, todos os pontos obtidos pelo cálculo dado pela bezier4.

De seguida define-se a geometria do tubo com o path previamente criado.

E o retorno da função é um objeto 3D tubular que será renderizado entre o ponto C0 e C1 tendo por base a equação quádrlica de Bézier.

createLineBalls – cria a partir de um ciclo for um array de linhas inicializado a zero.

updateLineBalls – atualiza a posição das linhas em função do movimento das bolas em qualquer eixo, a função é chamada quando se pressiona W ou S ou se move o rato.

resetScene – faz reset à cena, removendo todos os objetos e volta a reinseri-los.