E-fólio B | Instruções para a realização do E-fólio

UNIDADE CURRICULAR: Computação Gráfica

CÓDIGO: 21020

DOCENTE: António Araújo e Pedro Pestana

A preencher pelo estudante

NOME: Pedro dos Santos Gaspar

N.º DE ESTUDANTE: 1902551

CURSO: Licenciatura em Engenharia Informática

DATA DE ENTREGA: 03/01/2023

RELATÓRIO

O primeiro passo na realização deste e-fólio B, foi fazer o "scaffolding" das diretorias e dos ficheiros. Comecei por criar um ficheiro index.html similar ao do e-fólio A, e criei também a diretoria ./src com o ficheiro script.mjs. De seguida, criei o ficheiro bezier4.mjs e passei ao desenvolvimento da primeira parte do e-fólio: a implementação da função de cálculo de uma bézier quártica. Felizmente já tinha alguma experiência com curvas Bézier, pois são utilizadas em CSS para animações (se bem que em CSS são utilizadas curvas cúbicas e não quárticas).

Para implementar a função, bastou apenas seguir a equação da curva disponibilizada no enunciado e aplicá-la para obter os valores de x, y e z da curva no instante t. Extraí os valores constantes às 3 equações para variáveis, para evitar calcular estes valores 3 vezes desnecessariamente.

Com a função feita, passei ao desenvolvimento da página. Aqui, aproveitei algum do código que desenvolvi no e-fólio A, como algumas das variáveis globais, e a lógica de inicialização do plano, câmara, controlos, renderização, limpeza dos objetos. Alterei as cores dos ladrilhos do tabuleiro para ficarem mais transparentes, e adicionei uma linha que identifica o eixo z.

Após ter o plano funcional, adicionei as 5 bolas ao plano em posições x, y aleatórias, como dita o enunciado. Sempre que é feito reset ao plano (backspace), as bolas são novamente colocadas em posições aleatórias. Adicionei um *event listener* do movimento do rato, para gravar a posição do rato na janela, e no plano (código este desenvolvido também no e-fólio A). Adicionei ainda outro *event listener* para quando uma tecla é pressionada. Quando se pressiona um número, a respetiva bola fica com opacidade total. Ao pressionar no espaço, coloca a bola na última posição do rato no plano gravada. Pressionar novamente o número da bola selecionada, remove-a da seleção. Ao pressionar a tecla X, limpa o plano utilizando a função de limpeza adaptada do e-fólio A.

Aproveitei também este momento para criar a lógica para subir e descer a bola. Como o enunciado dita que o movimento inicialmente é de 0.1 unidades mas tem de ser possível mexer mais rápido se tiver pressionada a tecla, tive de implementar esta lógica de acelaração. O *KeyboardEvent* do JavaScript tem uma propriedade chamada *repeat* que indica se a tecla continua a ser pressionada. Utilizo esta propriedade para gravar qual a tecla que está a ser pressionada e por quantos "frames" foi pressionada. Esta quantidade é enviada para o método que sobe/desce a bola selecionada, e multiplica esta quantidade por 0.1, para obter o valor de quantas unidades deve subir/descer neste frame. Implementei também uma constante global para prevenir que a bola acelere para velocidades astronómicas, facilmente causando um overflow ou problemas de performance.

Ao subir e descer a bola, é validado também se a bola tem um valor de z diferente de 0. Se for o caso, desenha uma linha desde o centro da bola ao centro do referencial no eixo z.

Para indicar ao utilizador que bola está selecionada, assim como a atual posição das bolas e do rato, adicionei caixas no HTML e dei-lhes estilos com CSS. Estes estilos estão contidos no ficheiro ./styles/site.css. Ainda dentro desta diretoria está um ficheiro chamado normalize.css. Adicionei este ficheiro para, como o nome indica, "normalizar" os estilos dos elementos HTML que diferem entre os vários navegadores. Assim, independentemente do navegador utilizado, terá sempre uma experiência igual. Atribuí IDs a estas caixas para as referenciar no JavaScript. Sempre que o rato mexe, atualiza a caixa da posição do rato. Sempre que é selecionada ou movida uma bola, atualiza a caixa das informações das bolas. Aqui, podia ter utilizado o TextGeometry do Three.JS para escrever o texto, mas por simplicidade, e como o enunciado não menciona esta obrigatoriedade, foi mais fácil utilizar o HTML para tal. Aliás, a documentação do Three.JS até indica que esta é a forma mais fácil e rápida de introduzir texto.

Por último, resta desenhar efetivamente a curva de Bézier. Decidi criar uma classe num ficheiro à parte e exportá-la como módulo de JavaScript. Esta classe é uma adaptação do exemplo presente na página de documentação do TubeGeometry, sendo que recebe as coordenadas das 5 bolas, e utiliza a nossa função bezier4 para calcular a posição no instante t (fornecido como parâmetro na função getPoint do THREE.Curve). A cor de cada curva é calculada aleatoriamente.

Desenvolvi algumas funções utilitárias, e exportei-as como módulos, devidamente comentadas, no ficheiro ./src/helpers.mjs.

Testei e validei o funcionamento do site nas últimas versões dos navegadores Chrome, Edge e Firefox.