

”

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

UNIDADE CURRICULAR: Computação Gráfica

CÓDIGO: 21020

DOCENTE: António Araújo / Pedro Pestana

A preencher pelo estudante

NOME: Pedro dos Santos Gaspar

N.º DE ESTUDANTE: 1902551

CURSO: Licenciatura em Engenharia Informática

DATA DE ENTREGA: 21-11-2022

TRABALHO / RESOLUÇÃO:

Iniciei a realização deste e-fólio pelo desenvolvimento do algoritmo do ponto médio. Em primeiro lugar, tentei entender melhor o que é este algoritmo e para que serve.

Concluí que o que se pretendia era uma implementação do algoritmo de Bresenham. Ora, existem muitas implementações na internet deste algoritmo, nas mais variadas linguagens de programação, e obviamente que encontrei em JavaScript também, mas não me servia de nada copiar o algoritmo para o meu e-fólio e prosseguir. Tentei entender melhor como funciona, e como utiliza a variável de erro para aproximar os cálculos, mantendo a velocidade de execução. Tive de adaptar também o código para armazenar e retornar uma lista de coordenadas, e exportar a função como um módulo ECMAScript (extensão .mjs). Finalmente, dediquei algum tempo a fazer testes a esta função, para me certificar que devolve os resultados corretos, no tipo de dados pretendido no enunciado.

Na segunda parte do código, comecei por escrever o *boilerplate* HTML e utilizar o CSS necessário para a grelha aparecer corretamente (retirar os espaçamentos do *body*, para que não apareçam scrollbars). Na parte do JavaScript, como ainda não tinha muita experiência com o Three.JS, tentei resolver os problemas um a um à medida que surgiam, e dividir o trabalho em pequenas tarefas: em primeiro lugar, mostrar uma grelha única, de seguida dividir a grelha numa tela de 21x21 ladrilhos, depois colori-los de cores diferentes.

O próximo passo foi implementar o OrbitControls para permitir manipular o plano, assim como fazer zoom in e zoom out. Implementei os *event listeners* para deteção da posição do rato, o pressionar da tecla X e da tecla *Backspace* do teclado, e também do redimensionamento da janela para manter as dimensões do canvas sempre igual à altura e largura da janela do browser.

O passo mais complexo e onde demorei mais tempo foi a seleção de um ladrilho ao pressionar a tecla X e subsequente utilização do algoritmo do ponto médio, e para simplificar o código dividi esta parte em 3 funções. A primeira função, *onTileSelect*, trata de gerir e guardar quais os ladrilhos que foram pressionados, e ao escolher o ladrilho de fim, chama a função *fillMidpointTiles*, que pega no resultado do algoritmo do ponto médio (função *lineMP*), e preenche os ladrilhos com uma caixa amarela semi-transparente, com uma altura igual a $\frac{1}{4}$ do tamanho do ladrilho. Após preencher todas as caixas, é chamada a função *drawSelectionLine*, que desenha uma linha reta desde o ponto de início ao ponto de fim.

Por último, tratei do código de limpar a cena quando se clica no *Backspace*. Esta funcionalidade envolveu gravar todos os objetos criados na cena pelo utilizador, tal como as caixas amarelas e as linhas entre os ladrilhos, numa lista. Ao pressionar o *backspace*, todas estes objetos são removidos um a um da cena.

Para execução do código em ambiente de desenvolvimento, utilizei o pacote NPM live-server. O código foi validado pelo ESLint e tentei seguir a norma da nomenclatura de variáveis e funções (constantes globais upper snake case, variáveis locais e funções camel case).

Referências bibliográficas:

https://en.wikipedia.org/wiki/Bresenham%27s_line_algorithm

<https://threejs.org/docs/#manual/en/introduction/Installation>

<https://threejs.org/docs/#api/en/objects/Mesh>

<https://threejs.org/docs/#api/en/core/BufferGeometry>

<https://threejs.org/docs/#api/en/scenes/Scene>

<https://stackoverflow.com/questions/24437296/keeping-camera-above-ground-using-orbitcontrols-js>

<https://stackoverflow.com/questions/19426559/three-js-access-scene-objects-by-name-or-id>