

”

**E-fólio A** | Instruções para a realização do E-fólio

## COMPUTAÇÃO GRÁFICA | 21020

### Período de Realização

Decorre de 28 de outubro a **22** de novembro de 2022 (nota: prazo estendido; era até 21 de novembro)

### Data de Limite de Entrega

22 de novembro de 2022, até às 23:55 de Portugal Continental

### Trabalho a desenvolver:

Este trabalho consiste de duas partes: primeiro deverá implementar o algoritmo do ponto médio para desenhar segmentos de recta, como um módulo de JavaScript. De seguida deverá construir um interface gráfico de three.js que utiliza esse módulo para desenhar segmentos no ecrã, na forma de caixas (ladrilhos) sobre um plano num ambiente 3D. O plano representará um display raster virtual, onde poderá seleccionar pares de “pixels” com o mouse e o teclado, e desenhar segmentos entre esses pontos usando o algoritmo do ponto médio.

#### PARTE I- Implementação do algoritmo do ponto médio (2 valores):

Deverá implementar o caso geral do algoritmo do ponto médio na forma de um módulo de JavaScript.

Os seguintes parâmetros da implementação são obrigatórios (e eliminatórios):

- Deverá representar pontos por objectos literais de JavaScript na forma  $\{x: \text{int}, y: \text{int}\}$ . Por exemplo  $\{x:2, y:4\}$  representa o ponto (2,4). Os valores de x e y são sempre inteiros.
- O módulo tem que estar implementado num ficheiro chamado lineMP.mjs (note a extensão “mjs”) e esse módulo tem que exportar uma função chamada lineMP.
- Essa função lineMP tem como input um par de pontos e como output um array de pontos que corresponde ao resultado do algoritmo do ponto médio.
- O algoritmo tem que funcionar para qualquer par de pontos de coordenadas inteiras (sejam positivas ou negativas), dando a representação correcta do segmento de acordo com o algoritmo do ponto médio.

Exemplo:

```
let P = {x: 0, y: 0}; let Q = {x: 3, y: 1};
```

```
let R = lineMP(P,Q);
```

```
console.log(R); // imprime [{x: 0, y: 0}, {x: 1, y: 0}, {x: 2, y: 1}, {x: 3, y: 1}]
```

**Nota Importante:** é essencial que o algoritmo **corra** e que verifique as **especificações** do interface! Siga à risca as nomenclaturas e especificações apresentadas acima para as funções e objectos. O seu algoritmo vai ser testado por um programa que assume esses nomes antes mesmo do código ser inspecionado. Não serão avaliados módulos que não passem nesse teste ( que não corram, ou que não respeitem os parâmetros de input e output especificados ou que não façam a exportação correcta a partir do módulo).

Por exemplo, um ficheiro nomeado lineMP.js em vez de lineMP.mjs não será avaliado. Uma função lineMP que aceite pontos na forma de arrays em vez de objectos ([0,1] em vez de {x:0, y:1}) não será avaliado.

Os programas que corram e estejam adequadamente especificados serão alvo de avaliação de acordo com a sua correcção, eficiência, e elegância do código. Serão também avaliados pela documentação do mesmo.

## PARTE II- Interface Gráfico (2 valores)

Deverá fazer uma página web em three.js que funcione da seguinte forma:

### 1.

A página deverá correr a partir de um ficheiro index.html localizado na directoria de base do seu trabalho.

O código deverá ser importado a partir do index.html, na forma de módulos. **Preferencialmente** o código JavaScript deve estar totalmente contido em módulos e não inline no html. O html deve apenas importar e correr um módulo inicial, que chama os demais.

Um desses módulos terá por sua vez que importar o three.js e o OrbitControls. Essas importações deverão ser feitas a partir de CDN, da seguinte forma:

```
import * as THREE from 'https://unpkg.com/three@0.124.0/build/three.module.js';
```

```
import { OrbitControls } from  
'https://unpkg.com/three@0.124.0/examples/jsm/controls/OrbitControls.js'
```

É essencial que todas estas dependências funcionem (exceptuando claro os casos de eventuais indisponibilidades do servidor). **Só será alvo de avaliação código que corre.**

Deverá ainda importar o módulo lineMP.mjs, que desenhou na alínea acima para implementar o algoritmo do ponto médio. Este módulo deve estar na directoria base. Os outros módulos deverão estar numa directoria “./src”.

### 2.

A página deverá apresentar um plano dividido em quadrados coloridos (com cores alternadas, para ajudar a visualizar as divisões entre quadrados). Esse plano representa um *display raster*, sendo que cada quadrado representa um pixel. Além dos “pixels” devem estar visíveis os eixos positivos x e y, representados por linhas finas que atravessam o centro do pixel (0,0); o eixo positivo dos x é representado por uma linha azul e o positivo dos y por uma linha vermelha.

O “display” deve ter pelo menos 10 quadrados em cada uma das direcções (tanto para o lado positivo como para o lado negativo do pixel central, por isso terá pelo menos 21 quadrados de lado)

O utilizador deve conseguir controlar a vista usando os OrbitControls do three.js, com a vista centrada no centro do referencial (centro do pixel (0,0)).

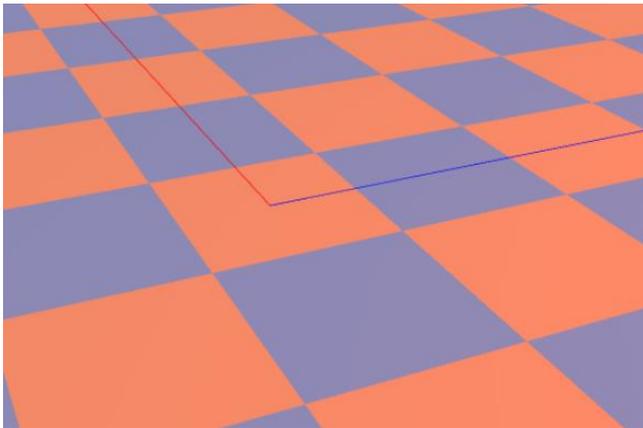
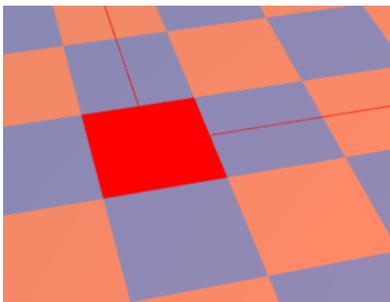


Fig 1: O “ecran” raster virtual, no estado inicial, com os pixels todos apagados.

Usando o *raycaster* do three.js o programa deve conseguir detectar o quadrado sobre o qual está o mouse nesse momento. O programa deve indicar para a consola a coordenada do pixel sobre o qual está em cada instante (só deve fazer log no momento em que muda de quadrado). Note que a coordenada em causa é a indexação do quadrado na grelha, e não necessariamente as suas “world coordinates”.

Quando o utilizador carregar na tecla “X” no teclado, o pixel da grelha em que o ponteiro do mouse se encontra deve mudar a sua cor para vermelho e as coordenadas do pixel na grelha devem ser armazenadas. Exemplo: Na imagem abaixo carregámos na tecla X quando o mouse estava por cima do pixel central. Guardou-se o ponto {x:0, y:0} neste caso e marcou-se o pixel a vermelho.



Quando de novo o utilizador carregar na tecla “X” do teclado, com o mouse por cima de um segundo pixel, acontece o seguinte:

- o segundo pixel fica vermelho; a sua coordenada de grelha é armazenada.
- o programa chama a função lineMP e calcula-a para os pontos correspondentes aos dois pontos seleccionados.
- O segmento resultante é renderizado com “tiles” (ladrilhos) assentes nos quadrados da grelha, representando os pixels acesos. Estes ladrilhos serão caixas semi-transparentes, amarelas, com uma altura igual a 1/4 do comprimento do lado dos quadrados.
- A linha **exacta** é também renderizada, a preto, ficando visível à transparência por baixo dos ladrilhos.
- É possível repetir o processo para desenhar mais linhas;
- Carregando no tecla Backspace apagamos todos os ladrilhos, restaurando a grelha inicial.

**EXEMPLO:** No exemplo da figura abaixo foram seleccionados pelo utilizador os pixels de coordenadas (0,0) e (3,1). Usando o lineMP acendemos os quatro pixels amarelos de coordenadas (0,0), (1,0), (2,1), (3,1) que correspondem ao algoritmo do ponto médio; Note-se que conseguimos ver à transparência tanto os eixos como a linha ideal que une o centro do pixel inicial (0,0) ao centro do pixel final (3,1). Note-se ainda à transparência a cor vermelha dos pixels inicial e final seleccionados, por baixo dos ladrilhos que representam os pixels acesos.

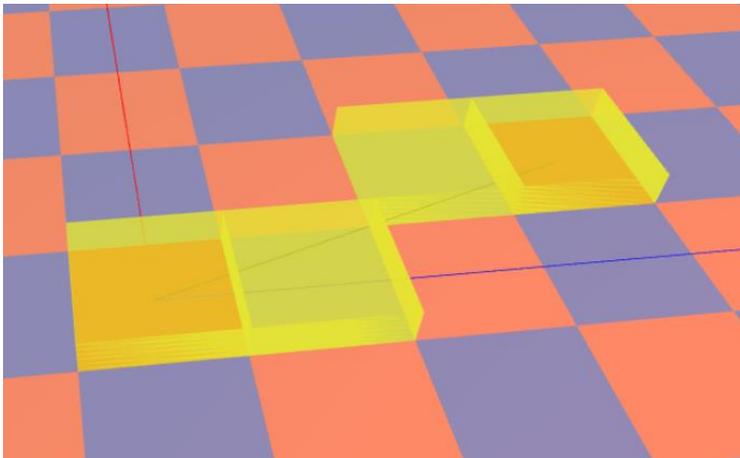


Fig. 2: Quatro pixels acesos. Note-se a linha exacta, visível à transparência.

Abaixo temos outras vistas da mesma cena, usando os OrbitControls para rodar/fazer zoom em torno do pixel central.

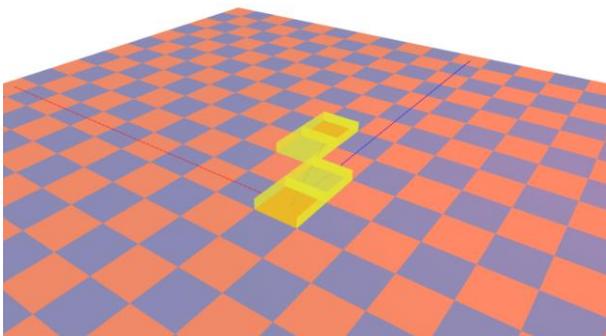


Fig. 3.

O OrbitControls permite obter uma vista de cima que é quase indistinguível da vista 2D (mas ainda é vista de perspectiva e não ortográfica):

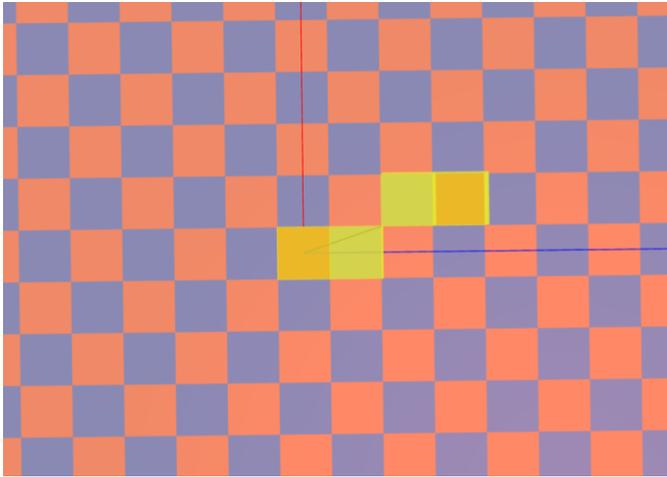


Fig. 4.

## Normas e indicações adicionais

Programa os algoritmos e métodos usando JavaScript, Html, CSS e Three js e comente adequadamente o código fonte.

Deve juntar um relatório explicativo de todas as decisões tomadas. Este relatório constituirá uma memória descritiva do trabalho realizado, devendo ser sucinto (máximo de 2 páginas A4, fonte 10, espaçamento simples) e complementar os comentários inseridos no código fonte;

Faça entrega de todos os ficheiros comprimindo-os num único ficheiro zip, adotando o seguinte formato: <Ultimo nome do aluno><numero de estudante>.zip.

Deve carregar o referido ficheiro para a plataforma no dispositivo E-fólio A até à data e hora limite de entrega. Evite a entrega próximo da hora limite para se precaver contra eventuais problemas.

Indique se detectar qualquer problema com o enunciado. Como o trabalho é individual, utilize o meu email em vez dos fóruns para qualquer dúvida cujo esclarecimento possa dar pistas impróprias aos seus colegas (na dúvida, assumo que é o caso): antonio.araujo@uab.pt

O ficheiro a enviar não deve exceder 8 MB.

Volto a frisar que o seu programa deverá estar **pronto para correr, sem dependências partidas**. Inclua na directoria base o ficheiro **lineMP.mjs**, o ficheiro **index.html** que corre a segunda parte do trabalho, e o relatório do trabalho (chamado ReadMe.pdf); quaisquer outros ficheiros, incluindo os seus outros módulos, eventuais ficheiros css, etc, devem ser colocados em pastas, sendo que os módulos JavaScript devem estar numa pasta “./src”.

O seu trabalho será avaliado por vários factores:

- a implementação das features pedidas.

- a correcção, elegância, e eficiência do código e da documentação do mesmo.

- **requisito essencial (eliminatório):** o programa tem que estar pronto a correr, sem alterações nem dependências quebradas, seguindo as normas estabelecidas acima para os interfaces e objectos.

Votos de bom trabalho!

A. Araújo