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

Computação Gráfica — 21020

Período de Realização
Consultar os prazos de entrega indicados pelos serviços.

Objetivos
O e-fólio global cobre potencialmente a totalidade da matéria lecionada.
A prova é composta por 5 questões (de onde serão escolhidas 4), contém 3 páginas e termina com a palavra **FIM**.

Recursos
A prova é individual, com consulta bibliográfica livre.

Critérios de Avaliação e cotação
Todas as respostas devem ser justificadas, salvo instrução em contrário. Respostas não devidamente justificadas são inválidas e terão cotação zero.
Todas as questões têm a mesma cotação, somando um total de 12 valores.

Normas as respeitar
Deve redigir o seu E-fólio na Folha de Resolução disponibilizada e preencher os dados do cabeçalho. A prova deve ser entregue como um único ficheiro pdf, com um máximo de 8 megabytes. Não são aceites outros formatos.
O nome do ficheiro deve ser: número de estudante seguido do seu apelido, seguido de EfolioG. Exemplo: 00000AraujoEfolioG.pdf
Utilize letra legível, se a prova for manuscrita. Atente à qualidade e legibilidade da digitalização.
No ato da entrega, assegure a integridade do ficheiro. Ficheiros que não abrem não podem ser corrigidos.
O e-fólio global dura 90 minutos, tendo uma tolerância de 60 minutos adicionais para digitalizar e carregar o ficheiro na plataforma.
Deve carregar o referido ficheiro para a plataforma no dispositivo disponibilizado para o efeito, até à data e hora limite de entrega. Evite a entrega próximo da hora limite para se precaver contra eventuais problemas técnicos.

Votos de bom trabalho!

António Araújo

Resolva apenas 4 das seguintes 5 questões.

1. Considere o algoritmo de Bresenham. Aplique-o graficamente para desenhar o segmento que une os pontos (0,5) e (2,0). Apresente a lista dos pontos que devem ser iluminados, e desenhe-os numa grelha. Nota: *não* é para executar o algoritmo algebricamente; deve executar o algoritmo de Bresenham geometricamente, ou seja, numa quadrícula (que pode fazer, ou usar uma já feita) deve desenhar o segmento e encontrar na quadrícula, visualmente, os pontos a iluminar, explicando com base no algoritmo porque escolheu esses pontos.

2. Considere o triângulo $\triangle ABC$ com $A = (4, 0, 10), B = (4, 0, 6), C = (4, 6, 6)$. Considere a rotação R que preserva o lado BC e transforma A em $A' = (8, 0, 6)$. Considere a projecção de perspectiva com centro de projecção em $(0, 0, 0)$ e plano de projecção $z = 2$. Apresente:

1. a matriz de R
2. a matriz P da projecção de perspectiva.
3. a matriz $C = PR$, composição de P com R.
4. a projecção em perspetiva de $A'BC$ por aplicação de C aos pontos de $\triangle ABC$.

3. Considere a curva de Bézier com os pontos de controlo $(0, 0)$, $(3, 2)$, $(9, 0)$.
a) Calcule os pontos da curva que correspondem a t = 0,2 e t = 0,8.
b) Represente graficamente a curva.

4. Nota: a presente questão será útil para poder obter cotação máxima na última questão desde enunciado (a questão de programação). Isto poderá ser relevante para a sua escolha.

a) Defina teoricamente o que é o *frustum* (ou volume de visualização) da camera de perspectiva e explique como se relaciona com os parâmetros de uma camera declarada em three.js. Ilustre com um diagrama se necessário.

b) Considere uma esfera S de raio R com centro na posição $(0, 0, d)$, com $d > R$, e um frustum com vértice em $(0, 0, 0)$. Dizemos que esse frustum é *minimal* relativamente à esfera S se os limites do frustum tocam tangencialmente a esfera. Ou seja, o frustum minimal é o o frustum mais pequeno que contém a esfera. Em particular, o viewport associado ao frustum minimal apresenta a esfera de tal forma que ela parece tocar exactamente um ponto em cada um dos lados do viewport (Figura 1), e o seu volume de corte elimina tudo o que esteja para a frente ou para trás da esfera.

Calcule os parâmetros do frustum minimal como função da distância d e do raio R. Explique a sua resposta na folha de exame, com a ajuda de um diagrama.

Nota 1: o problema é uma aplicação simples de trigonometria e do teorema de pitágoras. Pense qual tem que ser o ângulo de fov minímo para cobrir a esfera, e quais os limites anteriores e posteriores do frustum..

Nota 2: No problema final desta prova será útil escrever esta função algébrica que calculou como uma função javascript de dois floats d e R. Quando o fizer recorde que as potências em javascript funcionam com o operador ** e não com o operador ^. Por

exemplo `x**2` calcula x^2. Não se esqueça ainda que o fov em three.js vem em graus e não radianos, mas as funções trigonométricas geralmente usam radianos; recorde que PI radianos são 180 graus. Mais dicas: Em javascript a função arco tangente é `Math.atan`, raíz quadrada é `Math.sqrt`, a constante π é `Math.PI`, etc.

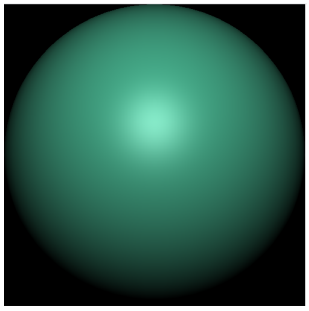

Figura 1: Uma esfera, vista no viewport do seu frustum minimal; a esfera toca os extremos do viewport.

5. Considere a seguinte cena a implementar em three.js. Uma esfera de raio R move-se com o seu centro C sobre o segmento de recta AB, em que $A = (0, 0, d)$ e $B = (0, 0, d+L)$. Há uma camera de perspectiva em (0,0,0) e uma point light um pouco por cima da camera. A esfera move-se sobre o segmento num loop interminável, para trás e para a frente entre A e B. O movimento deve ser sinusoidal com o tempo, ou seja, a coordenada z varia com uma função $cos(t)$ (`Math.cos(t)` em JavaScript).

A camera deve permanecer sempre no ponto $(0, 0, 0)$, mas mudar os seus parâmetros (fov, etc.) com o movimento da esfera de tal forma a que o seu frustum seja sempre o frustum minimal para a esfera (ver pergunta 4). Isto quer dizer que em todos os momentos o viewport parecerá tocar a esfera em quatro pontos (Figura 1), e que a esfera parecerá imóvel e de tamanho constante nesse viewport, apesar de estar a afastar-se a aproximar-se da camera (este movimento será apenas detectável pelo movimento que a luz reflectida fará na sua superfície da esfera). Ou seja, queremos uma camera dinâmica que mantenha sempre a esfera perfeitamente enquadrada durante o seu movimento, capturando a esfera "sem folgas".

Note-se que há aqui 3 parâmetros: três floats d, R e L. Estes poderiam ser inputs do utilizador. Aqui vamos defini-los apenas como consts no início do programa. Faremos, para concretizar, $d = 10, R = 5, L = 40$, mas o programa deverá funcionar (e eu testá-lo-ei) para quaisquer valores de d, R, e L (com $L > R$) que eu coloque nesses `const`.

Formato de entrega: : O código deve ser escrito directamente num único ficheiro html, que idealmente deverá correr sem erros (mas pode ter cotação parcial mesmo que não corra). Não inclua a library three.js nem ficheiros js auxiliares no seu zip, mas apenas o ficheiro index.html com o código inline, e que deverá chamar o three.js num repositório online, por exemplo assim:

```
<script type="module">
import * as THREE from 'https://unpkg.com/three@0.124.0/build/three.module.js';
```

Não deve depender se nenhuma outra library nem de assets auxiliares. No zip só vem a folha de exame e o html.

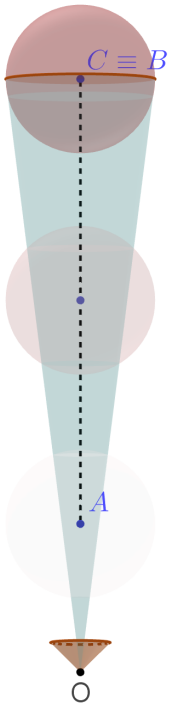

Figura 2: Uma esfera de centro C move-se em loop, para trás e para a frente, entre os pontos A e B; a camera mantém a esfera sempre enquadrada num viewport minimal (Figura 1).

Notas:

- Se não conseguir fazer tudo não desista! Há pontuação parcial por conseguir por a esfera no ecran, fazê-la mover-se, iluminá-la, etc.; se não conseguir movimento sinusoidal faça linear; Se não conseguir movimento faça estático e enquadre. Nesse caso explique na folha de enunciado ou em comentários as opções que tomou, e porquê.
- o programa deve funcionar para quaisquer valores de d, R, e L, mas se não conseguir, ao menos que enquadre o melhor possivel o movimento no caso standard dos valores definidos acima.
- Se não conseguiu obter o frustum minimal, ainda tem cotação se o frustum e viewport forem mais ou menos correctos. Pode ajustar a olho, correndo o código e modificando. A ideia é que a camera dever seguir e enquadrar a esfera em movimento o melhor possível. Mas tente fazer o frustum minimal, é bastante mais limpo e não é complicado.
- Note que quando faz update do fov deve fazer "camera.updateProjectionMatrix();" a seguir para que o resultado seja visível.
- poderá ser útil o comando lookAt;

FIM (Boa Sorte!)