(8150) Computação Gráfica e Interfaces

2017/11/03 FCT/UNL Duração: 1h30m

Teste nº1

1. **Classifique cada afirmação** como verdadeira (**V**) ou falsa (**F**). Uma resposta errada desconta 50% da sua cotação!

Considere o pipeline gráfico 3D do WebGL:

(A) `gl_PointSize` é uma variável the pode ser afetada com um valor no *vertex shader*.
(B) `gl_PointSize` tem que permanecer constante para cada chamada de desenho de primitivas (`drawArrays()` ou `drawElements()`).
(C) Um `attribute` consiste num pedaço de informação associada a um vértice pela aplicação.
(D) Uma variável `varying` recebe um valor atribuído pela aplicação javascript.
(E) Uma variável `uniform` pode ter o seu valor alterado, de vértice para vértice, durante uma mesma chamada de desenho de primitivas.
(F) Uma variável `uniform` pode ter o seu valor alterado, de fragmento para fragmento, durante uma mesma chamada de desenho de primitivas.
(G) `gl_FragColor` é uma variável que terá sempre que receber um valor (no fragment shader).
(H) `gl_Position` é uma variável que terá sempre que receber um valor (no vertex shader).
(I) As variáveis `uniform` de cada um dos shaders dum programa GLSL são independentes das definidas no outro shader.
(J) As variáveis `varying` dos dois shaders dum mesmo programa GLSL têm que ser exatamente as mesmas.

2. **Escolha a opção correta**. Uma resposta errada desconta 25% da sua cotação!

Uma aplicação javascript que armazene, quer a posição, quer a cor de cada vértice:
(A) Terá que as guardar num único buffer e estas não poderão estar dispostas alternadamente no mesmo.
(B) Terá que as guardar num único buffer mas estas poderão estar guardadas em regiões separadas ou de forma alternada.
(C) Pode guardá-las em buffers separados ou num mesmo buffer, tendo neste caso que ser guardadas em regiões separadas, não podendo ser dispostas alternadamente.
(D) Pode guardá-las em buffers separados ou num mesmo buffer, podendo ainda, neste caso, ser armazenadas em zonas separadas ou de forma alternada.

3. **Classifique cada afirmação** como verdadeira (**V**) ou falsa (**F**). Uma resposta errada desconta 50% da sua cotação!

Uma aplicação tem uma cena composta por um ou mais objetos 3D e desenha-os usando apenas faces, mas usando apenas um dos modos de desenho diretamente suportados pelo sistema gráfico: malha de arame (apenas se desenham as arestas de cada face) ou superfícies preenchidas (pintando o interior de cada face poligonal). Uma cena nunca será desenhada misturando estes dois modos.

Considere o problema da remoção de linhas e superfícies ocultas e as técnicas estudadas - método do produto interno (face culling) e algoritmo z-buffer..

(A) O z-buffer opera face a face descartando num único passo aquelas que estão totalmente ocultas.

(B) O z-buffer pode ser usado isoladamente quer em modo wireframe, quer em modo de preenchimento, para qualquer cena.
(C) O z-buffer necessita ser limpo a cada quadro (frame) se os objetos forem animados.
(D) Qualquer implementação do z-buffer guarda sempre, para cada pixel, a distância à câmara.
(E) Qualquer implementação do z-buffer necessita dum buffer onde se armazena um valor por pixel.
(F) O método do produto interno (face culling) resolve o problema da visibilidade para ambos os modos de desenho (wireframe e preenchimento) para qualquer cena com apenas um objeto.
(G) O método do produto interno (face culling) não resolve o problema da visibilidade para cenas compostas apenas por poliedros convexos (plural) mas contribui para a aceleração do seu desenho ao descartar faces completamente ocultadas.
(H) Numa cena com um único poliedro convexo é necessário usar ambos os métodos (face culling + z-buffer) para resolver o problema da visibilidade no modo de desenho com preenchimento.
(I) Numa cena com vários poliedros convexos poder-se-á usar apenas o método do produto interno para ambos os modos de desenho produzindo os resultados corretos em termos de visibilidade.
(J) Uma aplicação carrega modelos de objetos a partir dum ficheiro contendo estes uma tabela de vértices e uma tabela de faces. Para estes modelos a aplicação consegue computar as normais viradas para o exterior do objeto se todas as faces forem especificadas usando uma mesma orientação (positiva ou negativa) sabendo a aplicação qual a orientação que foi usada para as especificar.

4. Escreva o código dos shaders duma aplicação que irá mostrar um conjunto de objetos 3D usando um efeito de transparência. O efeito irá aumentar a transparência das superfícies dependendo da sua coordenada z, medida num determinado referencial (sistema de coordenadas). Se a distância for abaixo de 0,5 unidades, os objetos serão totalmente opacos. Entre 0,5 e 0,8 unidades a transparência aumenta sendo totalmente transparentes para distâncias em z superiores a 0,8 unidades.
   Cada vértice dos objetos a desenhar tem uma posição 3D (em coordenadas homogéneas) dada no referencial do mundo (WC) e uma cor RGB. A aplicação envia ainda ao programa GLSL a matriz **M** que transforma de coordenadas do mundo para as coordenadas do referencial a usar para o efeito de transparência. Preencha os espaços na folha de resposta por forma a ter um programa GLSL válido e capaz de implementar o efeito pretendido.

5. Uma aplicação para ler e-books necessita mostrar simultaneamente o conteúdo de duas páginas A4 (210 x 297 mm), lado a lado, ao alto, centradas horizontalmente e alinhadas com o topo do ecrão, sem qualquer espaço entre elas. O dispositivo tem uma resolução de 1920x1080 pixels (16:9) e a sua origem localiza-se no canto superior esquerdo. Neste dispositivo deverá ser deixada, em baixo, uma margem de 180 pixels de altura para mostrar as miniaturas das páginas. O conteúdo de cada página do livro é descrito usando um sistema de coordenadas localizado no canto inferior esquerdo da página e cujas unidades são dadas em milímetros (mm). A solução deverá maximizar o espaço de ecrã usado e sem deformação nem recorte.

   A. Determine os formatos (relações de aspeto) da janela e das áreas disponíveis para os visores bem como as dimensões finais dos visores. Não efetue qualquer cálculo, bastando indicar as operações necessárias para determinar as dimensões caso venham a ter que ser calculadas.

   B. Especifique a transformação de enquadramento, **M**, que será usada para o caso da página da esquerda como uma sequência de transformações geométricas elementares (**T**, **R** or **S**), com indicação dos respetivos parâmetros, a ser usada na forma **P'**=**M**.**P**.

   C. O mesmo que em B), mas agora para a página da direita (visor direito)

   D. Imagine que necessita implementar um mecanismo de selecção (picking) no espaço dos objetos. Apresente a transformação que seria necessária para passar de coordenadas do ecrã para coordenadas do objeto (mundo) para o caso do visor da esquerda (página da esquerda).

Nº: _____ (8150) CGI - Teste nº1 - 2017/11/03 Sala: _____

Nome: _____

1. Preencha cada coluna com um 'X' para assinalar a sua resposta.

	(A)	**(B)**	**(C)**	**(D)**	**(E)**	**(F)**	**(G)**	**(H)**	**(I)**	**(J)**
V (verdadeiro)										
F (falso)										

2. Resposta: _____

3. Preencha cada coluna com um 'X' para assinalar a sua resposta.

	(A)	**(B)**	**(C)**	**(D)**	**(E)**	**(F)**	**(G)**	**(H)**	**(I)**	**(J)**
V (verdadeiro)										
F (falso)										

4. Preencha os espaços em branco.

```
uniform mat4 M;
_________ ____ p;
_________ vec3 c;
_______ _____ f;
_______ float d;

void main() {
  __________ = p;
  __ = c;
  d = (__ * __).z;
}
```

```
_______ _____ f;
_______ _____ d;

void main() {
  float a = smoothstep(__,__,d);
  ___________ = vec4(___, ___);
}
```

5. Preencha as caixas com a sua resposta!
   5.A.

5.B.

5.C.

5.D.

Boa sorte!