

**Normativa preguntes curtes**

1. Responen les següents preguntes en el mateix full de l'enunciat.
2. Cal que les respostes siguin **clares, precises i concises**.
3. No es poden usar apunts ni calculadores ni cap dispositiu electrònic.

**Escena 1:** Tenim una escena formada per un terra quadrat de 20x20 ubicat en el pla XZ, centrat a l'origen de coordenades i amb costats paral·lels als eixos de coordenades, dos Homers d'alçada 4 i 8 ubicats amb el centre de la base de la seva capsa mínima contenidora en els punts (-8, 0, 0) i (8, 0, 0) respectivament, mirant en direcció X- i un Patricio d'alçada 10 amb el centre la base de la seva capsa en (0, 0, 7.5) i mirant en direcció Z+. Tant el Patricio com els Homers estan escalats uniformement. Els VAOs del terra, del Homer i del Patricio ja estan creats (VAO\_terra, VAO\_Homer, VAO\_Patricio) i només el VAO\_terra conté un VBO amb les posicions dels vèrtex ja en sistema de coordenades de l'aplicació.

1. (1 punt) Tenint en compte que volem pintar l'**Escena 1**, resol els següents apartats amb el pseudo-codi adient:

- a) Sabem que els punts mínim i màxim de la capsa contenidora del model del Homer són: (Hminx, Hminy, Hminz) i (Hmaxx, Hmaxy, Hmaxz), i que el model mira inicialment en direcció Z+. Completa la funció següent que calcula i retorna la TG necessària per pintar cadascun dels dos Homers de l'escena. Aquesta funció rep com a paràmetres la posició final del centre de la base del Homer i l'alçada que aquest ha de tenir.

```
glm::mat4 TG_Homer(pos, alçada)
{
    TG = I;

    ...

    return (TG);
}
```

- b) Suposant que també tenim la funció TG\_Patricio() que ens retorna la TG necessària per pintar el Patricio de l'**Escena 1**, la funció modelTransform(TG) que envia una TG al Vèrtex Shader, i que podem pintar el model amb la funció pintaModel(model) on el model el podem indicar com Homer, Patricio o Terra, indica el pseudo-codi necessari per a que es pinti l'**Escena 1** al complet. Utilitza també la funció definida a l'apartat a).

2. (1 punt) Tenint en compte que els models de l'**Escena 1** tenen definits colors per vèrtex en el seus corresponents VBOs, completa correctament el codi del Vertex Shader i del Fragment Shader següents per a que es pugui visualitzar correctament l'escena amb colors, des d'una càmera prèviament passada als shaders, i amb els objectes posicionats correctament.

Vertex Shader:

```
#version 330 core

.....   vec3 vertex;
.....   vec3 color;

.....   mat4 TG, View, Proj;

.....   vec3 fcolor;

void main() {
    .....
    gl_Position = ..... * vec4(vertex,1);
}
```

Fragment Shader:

```
#version 330 core

.....   vec3 fcolor;

.....   vec4 FragColor;

void main () {
    FragColor = .....
}
```

3. (1 punt) Tenint en compte l'**Escena 1**, completa tots els paràmetres d'una càmera en perspectiva que permeti veure en 3<sup>a</sup> persona l'escena sencera, centrada en un Viewport de 600\*300 píxels, sense deformacions, desaprofitant el mínim d'espai possible, i ajustant al màxim els plans de retallat. Considera que els angles estan en graus.

```
VM = lookAt ( (      , 5, 0) , (0,      , 0) , (0, 1, 0) );

PM = perspective ( 60 ,      ,      ,      );
```

4. (1 punt) En una aplicació de disseny gràfic es genera un dibuix que té 4 colors, codificats en RGB com: C1 = (1, 0.5, 0), C2 = (1, 0, 0.8), C3 = (0.5, 0, 0.86) i C4 = (0, 1, 1).
- Quan el dibuixant envia a imprimir el dibuix en una impressora CMY i usant paper blanc, els colors C1 i C2 li surten bé, però els C3 i C4 no li surten correctament. Podries dir què li passa a la impressora?
  - Suposant que la impressora funciona correctament, indica de quin color es veurà la part del dibuix del color C2 si aquest s'imprimeix usant paper groc de màxima intensitat.
  - Un cop imprès el dibuix sense problemes i usant un paper blanc, el dibuixant intenta mirar-se'l a través d'un filtre de color groc (només deixa passar llum groga). De quins colors es veurà el dibuix? Indica la codificació RGB dels quatre colors del dibuix vistos a través del filtre.
  - Tenint en compte els colors inicials del dibuix, si pensem en la seva representació en format HSB, hi ha una de les components (H, S o B) que té el mateix valor en tots ells. Indica quina és aquesta component i quin valor té.

Nom i cognoms:

**Normativa del test**

- (a) A les graelles que hi ha a continuació, marca amb una creu les teves respostes de l'examen. **No es tindrà en compte cap resposta fora d'aquestes graelles.**
- (b) No es poden usar apunts, calculadores ni cap dispositiu electrònic.
- (c) Totes les preguntes tenen una única resposta correcta.
- (d) Les preguntes contestades de forma errònia tenen una **penalització del 33%** del valor de la pregunta.

Num	A	B	C	D
5				
6				
7				
8				

Num	A	B	C	D
9				
10				
11				
12				

Num	A	B	C	D
13				
14				
15				
16				

5. (0.5 punts) La View Matrix serveix per a:

- a) Transformar els vèrtexs del sistema de coordenades de l'Aplicació al sistema de coordenades de l'Observador.
- b) Transformar els vèrtexs del sistema de coordenades de Model al sistema de coordenades de l'Observador.
- c) Transformar els vèrtexs del sistema de coordenades de l'Observador al sistema de coordenades de l'Aplicació.
- d) Transformar els vèrtexs del sistema de coordenades de l'Observador al sistema de coordenades de Clipping.

6. (0.5 punts) Definim una càmera amb OBS = (3,3,3) i VRP = (0,0,0). Quin dels següents vectors UP seria vàlid?

- a) UP = (1,1,1)
- b) UP = (-2,-2,-2)
- c) UP = (1,0,0)
- d) Tots són correctes

7. (0.5 punts) Si utilitzem un degradat de color per al fons d'una interfície on els botons són tots del mateix color...

- a) És possible que es percebin els botons amb diferent intensitat.
- b) És recomenable de cara a dissenyar interfícies per a personas daltòniques.
- c) Ajuda a millorar el contrast entre botons i fons.
- d) Hauríem d'escollir sempre colors saturats.

8. (0.5 punts) El Fragment Shader s'encarrega de...
- Processar tots els vèrtexs que hi arriben en el sistema de coordenades de clipping.
  - Processar tots els vèrtexs que hi arriben en el sistema de coordenades de dispositiu.
  - Processar tots els fragments que son rasteritzats assignant-los un color de sortida o descartant-los.
  - Processar tots els fragments que son rasteritzats i assignant-los a tots un color de sortida.
9. (0.5 punts) La capsula contenidora d'una escena té com a punt mínim el punt  $MIN = (-2, -2, 3)$  i com a punt màxim el punt  $MAX = (2, 2, 4)$ . Suposant que tenim una òptica perspectiva correctament definida, amb quina de les següents definicions de càmera en assegurarem de poder veure l'escena sencera?
- $OBS = (0, 7, 3.5)$ ;  $VRP = (0, 2, 3.5)$ ;  $UP = (0, 1, 0)$ ;
  - $OBS = (0, 8, 3.5)$ ;  $VRP = (0, 2, 3.5)$ ;  $UP = (0, 0, 1)$ ;
  - $OBS = (0, 0, 3.5)$ ;  $VRP = (0, 0, 0)$ ;  $UP = (0, 1, 0)$ ;
  - $OBS = (1.5, 0, 3.5)$ ;  $VRP = (0, 0, 3.5)$ ;  $UP = (0, 0, 1)$ ;
10. (0.5 punts) Si no ajustem bé els plans de retallat  $Z_{near}$  i  $Z_{far}$  a l'escena que volem visualitzar...
- Pot ser que es retalli contingut de l'escena.
  - Perdem precisió en el càlcul de les  $Z$  respecte al sistema de coordenades de dispositiu.
  - Pot ser que desaprofitem molt espai del nostre volum de visió perquè no hi haurà res a visualitzar.
  - Totes són correctes.
11. (0.5 punts) Tenim un Viewport que fa 300 píxels d'ample i 500 píxels d'alçada, i estem definint una càmera amb òptica ortogonal amb un Window que va del punt  $(-2.5, -1.5)$  al punt  $(2.5, 1.5)$ . Quina de les següents afirmacions és certa assumint que hem posat la resta de paràmetres de manera que es pugui veure tota l'escena?
- Veurem l'escena deformada.
  - Veurem l'escena retallada per els costats.
  - Veurem l'escena retallada per dalt i per baix.
  - Veurem l'escena correcta (sense retallar ni deformar).
12. (0.5 punts) Volem definir una View Matrix fent servir els angles d'Euler  $(\theta, \Psi, \varphi)$  per als eixos  $X, Y$  i  $Z$  respectivament (en el codi els teniu com `angX`, `angY` i `angZ` respectivament), un  $VRP$  i una distància  $d$  entre el  $VRP$  i l'observador. Quin dels següents pseudo-codis és el correcte?
- ```

VM = Translate(0, 0, d)
VM = VM * Rotate(-angZ, 0, 0, 1)
VM = VM * Rotate(angX, 1, 0, 0)
VM = VM * Rotate(-angY, 0, 1, 0)
VM = VM * Translate(VRP.x, VRP.y, VRP.z)
viewMatrix(VM)

```

- b) `VM = Translate(0, 0, -d)`  
`VM = VM * Rotate(-angZ, 0, 0, 1)`  
`VM = VM * Rotate(angX, 1, 0, 0)`  
`VM = VM * Rotate(-angY, 0, 1, 0)`  
`VM = VM * Translate(-VRP.x, -VRP.y, -VRP.z)`  
`viewMatrix(VM)`
- c) `VM = Translate(-VRP.x, -VRP.y, -VRP.z)`  
`VM = VM * Rotate(-angZ, 0, 0, 1)`  
`VM = VM * Rotate(angX, 1, 0, 0)`  
`VM = VM * Rotate(-angY, 0, 1, 0)`  
`VM = VM * Translate(0, 0, -d)`  
`viewMatrix(VM)`
- d) `VM = Translate(0, 0, d)`  
`VM = VM * Rotate(-angY, 0, 1, 0)`  
`VM = VM * Rotate(angX, 1, 0, 0)`  
`VM = VM * Rotate(-angZ, 0, 0, 1)`  
`VM = VM * Translate(-VRP.x, -VRP.y, -VRP.z)`  
`viewMatrix(VM)`

13. (0.5 punts) Quin és el mínim nombre de vèrtexs que necessites definir per poder dibuixar un quadrat amb OpenGL?

- a) 6
- b) 8
- c) 4
- d) Cap de les altres.

13.2 (0.5 punts) Quants VAOs i VBOs es necessiten per tenir ben definida l'**Escena 1** tenint en compte que per a cada vèrtex tenim informació de les seves coordenades i el seu color.

- a) VAOs = 2; VBOs = 2.
- b) VAOs = 2; VBOs = 4.
- c) VAOs = 3; VBOs = 6.
- d) Cap de les altres.

14. (0.5 punts) La diferència entre tenir una topologia implícita i una topologia explícita és:

- a) En una topologia implícita definim les cares fent referència als índexs d'una taula de vèrtexs i no tenim vèrtexs repetits, mentre que en una topologia explícita només tenim una taula amb repetició de vèrtexs i les cares venen definides per l'ordre d'aquests.
- b) En una topologia implícita definim les cares fent referència als índexs d'una taula de vèrtexs amb coordenades entre -1 i 1 en tots els eixos, mentre que en una topologia explícita els vèrtexs estan definits en una taula en coordenades de model i amb repetició de vèrtexs.
- c) En una topologia explícita definim les cares fent referència als índexs d'una taula de vèrtexs i no tenim vèrtexs repetits, mentre que en una topologia implícita només tenim una taula amb repetició de vèrtexs i les cares venen definides per l'ordre d'aquests.
- d) En una topologia explícita definim les cares fent referència als índexs d'una taula de vèrtexs en coordenades de l'aplicació, mentre que en una topologia implícita els vèrtexs estan definits en coordenades de model.

15. (0.5 punts) La inconsistència induïda consisteix en:

- a) Transmetre marca diferenciant-se de la competència.
- b) Fer els objectes diferents si es comporten de diferent manera als anteriors.
- c) Evitar que la sensació de l'usuari en mòbil sigui exactament igual a la d'un portàtil.
- d) Facilitar l'aprenentatge per a usuaris familiaritzats amb la marca.

16. (0.5 punts) El principi de Chunking consisteix en:

- a) Organitzar les dades seguint algun ordre, per exemple alfabètic.
- b) Mostrar a l'usuari interfícies amb agrupacions de pocs elements.
- c) Mostrar a l'usuari petites quantitats d'informació que pugui recordar uns segons.
- d) En web, quan l'usuari ha d'escollir entre diferents opcions, intentar mostrar al voltant de 5+/-2 alternatives.