

Interacció i Disseny d'Interfícies

Data: 19 d'abril de 2022

CONTROL PARCIAL

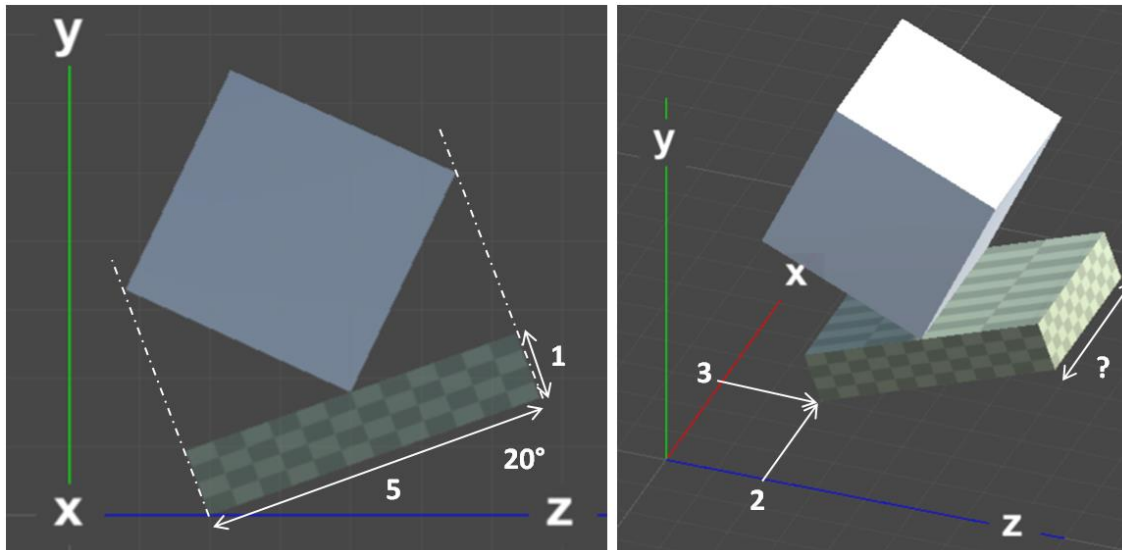
Temps: 2h

Feu cada problema en un full separat.
Deixeu les operacions indicades.

(1) [3,5p] A partir de la funció **pintaCub()**, que dibuixa un cub de **costat 1** centrat a l'origen, volem dibuixar l'escena mostrada a continuació:

A.- Base quadriculada: Prisma d'alçada 1, la base és rectangular, un dels costats de la base mesura 5 (vegeu dibuix). Una aresta reposa sobre el pla $y=0$.

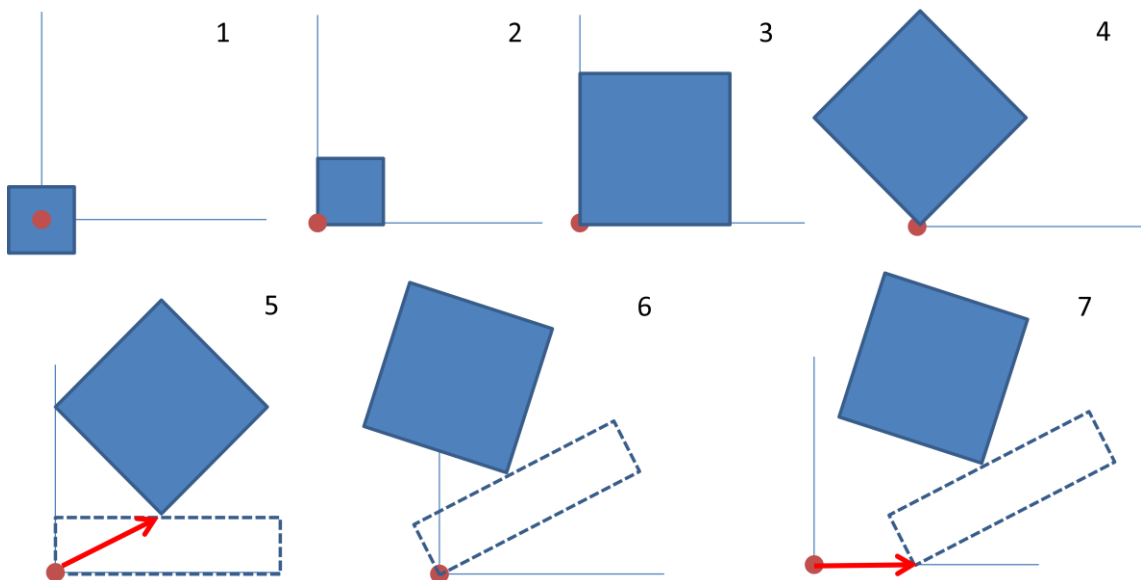
B.- Cub blanc: Cub centrat sobre A, en equilibri perfecte sobre la seva aresta. Les cares inferiors del cub formen 45° amb la base. L'aresta del cub que toca al prisma quadriculat té la mateixa mida que la cara on es recolza.



Escriu les transformacions geomètriques necessàries per passar del cub original a les figures A i B usant les funcions de la llibreria glm. Fes-ho programant les funcions **modelTransformCubX()**, on X és l'etiqueta del fragment (A o B). La funció ha de crear i enviar la matriu. Useu codi el més semblant possible al C++ real.

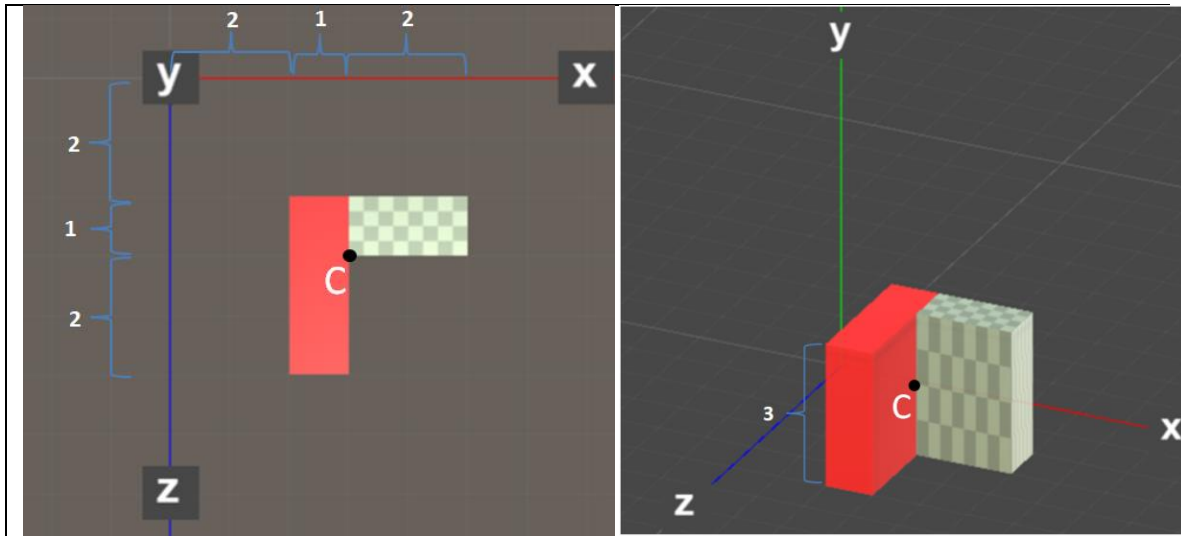
```
void modelTransformCubA() {  
    float diagonal = 5;  
    float c = diagonal/sqrt(2);  
  
    glm::mat4 TG(1.0f);  
  
    TG = glm::translate(TG, glm::vec3(3, 0, 2));  
    TG = glm::rotate(TG, glm::radians(-20), glm::vec3(1, 0, 0));  
    TG = glm::scale(TG, glm::vec3(c, 1, 5));  
    TG = glm::translate(TG, glm::vec3(0.5, 0.5, 0.5));  
    glUniformMatrix4fv (transLoc, 1, GL_FALSE, &TG[0][0]);  
}
```

Pel cub B el més fàcil era fer la rotació en dos passos, de manera que ens estalviem tota la trigonometria. Ajuda imaginar on està la base per fer tot el procés (en el dibuix està en línia discontinua)



```
void modelTransformCubB() {  
    glm::mat4 TG(1.0f);  
  
    float diagonal = 5;  
    float c = diagonal/sqrt(2);  
  
    TG = glm::translate(TG, glm::vec3(3, 0, 2));  
    TG = glm::rotate(TG, glm::radians(-20), glm::vec3(1, 0, 0));  
    TG = glm::translate(TG, glm::vec3(0.0, 1, 2.5));  
  
    TG = glm::rotate(TG, glm::radians(45.f), glm::vec3(1, 0, 0));  
    TG = glm::scale(TG, glm::vec3(c, c, c));  
    TG = glm::translate(TG, glm::vec3(0.5, 0.5, 0.5));  
  
    glUniformMatrix4fv (transLoc, 1, GL_FALSE, &TG[0][0]);  
}
```

(2) Una escena està formada per prismes adjacents (un pintat llis i l'altre quadriculat) estan disposats tal i com es mostra a la figura següent:



(2.1) [1,5p] Calcula els paràmetres de la projecció en perspectiva si volem orbitar l'escena amb euler angles a una distància del centre de l'escena de 10. El viewport on visualitzem l'escena té resolució 300x500.

Com que volem poder orbitar per l'escena, ens cal aproximar-la per una esfera. El primer pas és calcular la capsa contenidora de l'escena, i a partir d'aquí obtenir el radi de l'esfera contenidora.

$$RA_v = 300/500 = 0.6$$

$$p_{\text{Min}} = (2, 0, 2); p_{\text{Max}} = (5, 3, 5)$$

$$|p_{\text{Max}} - p_{\text{Min}}| = |(3, 3, 3)| = \sqrt{3^2 + 3^2 + 3^2} = 3 \cdot \sqrt{3} = D$$

(D és la diagonal de la capsa contenidora)

$$\text{Radi} = \frac{D}{2} = \frac{3 \cdot \sqrt{3}}{2}$$

$$z_F = 10 + \text{Radi} = 12.6$$

$$z_N = 10 - \text{Radi} = 7.4$$

Segons la fórmula vista a teoria, obtenim l'angle d'apertura (assumint un frustrun tangent a l'esfera)

$$\alpha = \arcsin\left(\frac{R}{d}\right) = \arcsin\left(\frac{\frac{3 \cdot \sqrt{3}}{2}}{10}\right) = 15.058 \text{ graus}$$

Com que la $RA_v < RA_w$ (si aproximem per una esfera $RA_w=1$), cal reduir el RA_w , això ho aconseguim augmentant l'alçada. Recalculem l'apertura vertical amb la fórmula:

$$\alpha^* = \text{atan} \left(\tan(\alpha) / RA_v \right) = \text{atan} \left(\tan(15.058) / 0.6 \right) = 24.15$$

Finalment obtenim FOV^* :

$$FOV^* = 2 \alpha^* = 48.30 \text{ graus}$$

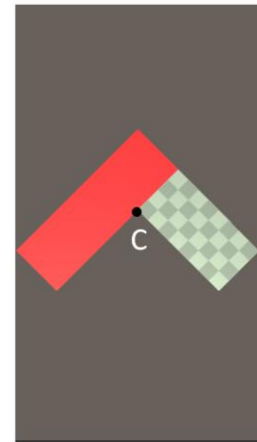
Solució:

$$FOV = 48.30 \text{ graus}, RA = 0.6, zN = 7.4, zF = 12.6$$

(2.2) Volem visualitzar ara l'escena ajustada **exactament** tal i com es mostra a la figura de la dreta, en un viewport de 300x500.

La figura ha de quedar centrada en vertical respecte el punt C (en direcció Y, està a la meitat de l'aresta).

La càmera s'ha de situar a una distància de 5 de C, i els plans de tall z_{Near} i z_{Far} han de ser el més ajustats possible.



(a) [0,5p] Calcula **tots els paràmetres** de la View Matrix.

$$VRP = (3, 1.5, 3)$$

$$OBS = (3, 6.5, 3)$$

$$UP = (-1, 0, -1)$$

(b) [2,5p] Calcula **tots els paràmetres** de la Projection Matrix per una càmera amb projecció perspectiva.

$$zN = 6.5 - 1.5 - 1.5 = 3.5$$

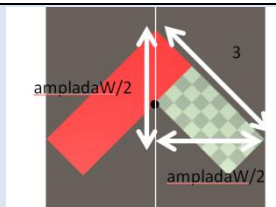
$$zF = zN + 3 = 6.5$$

$$RA_v = 300/500 = 0.6$$

La RA_w és major a RA_v (es pot fer el càlcul però al dibuix es veu clarament que la proporció de l'escena és clarament més quadrada que el viewport. Cal reduir RA_w , per tant hem d'augmentar l'alçada de la finestra, i per tant el FOV.

En aquest cas l'escena no s'aproxima per una esfera, sinó que és una vista "a mida", i per tant haurem de fer els càlculs de les dimensions de la finestra de projecció.

Primer calculem l'amplada de l'escena, doncs la finestra està encaixada en horitzontal. Apliquem Pitagores en el triangle marcat a la figura:



Obtenint:

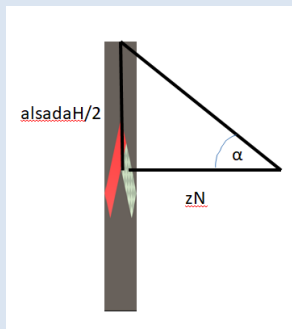
$$\text{ampladaW}/2 = \frac{3}{\sqrt{2}}$$

Calculem l'alçada de la finestra a partir de l'amplada i l'RAv

$$\text{alsadaH}/2 = \text{ampladaW}/2 / \text{RAv} = \frac{3}{0.6\sqrt{2}} = 5 \frac{\sqrt{2}}{2}$$

Calculem el FOV amb aquesta alçada:

$$\text{FOV} = 2 \cdot \text{atan}(\text{alsadaH}/2 / zN)$$



$$\text{FOV} = 2 \cdot \text{atan}\left(\frac{\sqrt{2}}{0.4 \cdot 3.5}\right) = 90.57 \text{ deg}$$

Finalment, cal forçar RAw igual a RAv:

$$\text{RAw} = \text{RAv} = 0.6$$

Resumint, els paràmetres de projecció són:

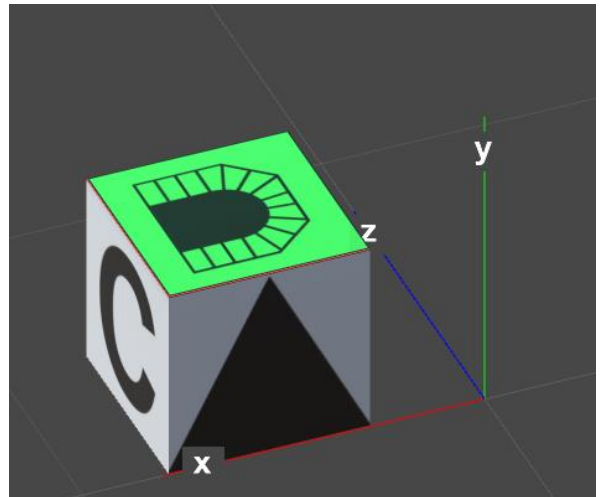
$$zN=3.5, zF=6.5, \text{FOV}=90.57, \text{RAw}=0.6$$

(3) [2p] A continuació s'indiquen diferents càlculs que usen transformacions geomètriques per obtenir una ViewMatrix (VM).

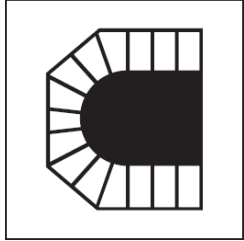

Aquestes VM s'utilitzarà per visualitzar una escena formada per un cub de costat 5 centrat a (5, 2.5, 2.5), amb els costats pintats amb els patrons que es poden veure a la figura de la dreta.

Per cada definició de VM, es demana que dibuixeu què veuria la càmera.

VIGILEU el que es veu a dreta i esquerra!



Nota: per resoldre aquest problema només calia aplicar les ViewMatrix a l'escena i ser conscient de que la càmera per defecte està a l'origen de coordenades (0,0,0) i que mira cap a z negativa i amb vector UP (0,1,0). S'havia de dibuixar l'escena des d'aquest punt de vista.

	Dibuix de la vista de càmera
<pre>glm::mat4 VM(1.0f); VM = glm::translate(VM, vec3(-5, 2.5, -10)); VM = glm::rotate(VM, radians(90), vec3(1, 0, 0));</pre>	
<pre>glm::mat4 VM(1.0f); VM = glm::translate(VM, vec3(0, 0, -15)); VM = glm::rotate(VM, radians(90), vec3(0, 0, 1)); VM = glm::rotate(VM, radians(-90), vec3(0, 1, 0)); VM = glm::translate(VM, vec3(-5, -2.5, -2.5));</pre>	

```
glm::mat4 VM(1.0f);
VM = glm::rotate(VM ,radians(-90),vec3(0,1,0));
```

No es veu, queda darrera de la
càmera.

```
glm::mat4 VM(1.0f);
VM = glm::translate(VM, vec3(0,0,-15));
VM = glm::rotate(VM ,radians(180),vec3(1,0,0));
VM = glm::translate(VM, vec3(-5,-2.5,-2.5));
```

