Interacció i Disseny d'Interfícies. Control parcial

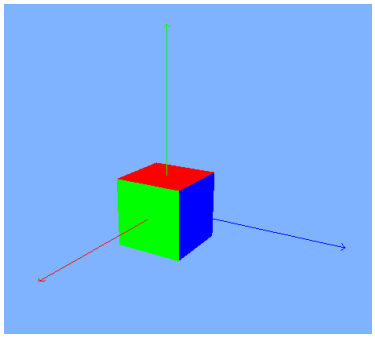
Dept. de Ciències de la Computació
E.P.S.E.V.G., 1 d'abril de 2020, 10:30-11:45

L'examen es lliurarà en format **pdf**. Useu LibreOffice Writer o MsWord per generar-lo. Si heu d'incloure dibuixos i no voleu usar una eina de dibuix, podeu fer la foto i enganxar al document.

Lliurament : Via tasca d'Atenea.
Format: Feu un ZIP anomenat Cognom1Cognom2_Nom.zip que contingui:

- Enunciat.pdf
- Solucio.pdf

Disposem d'un VAO que defineix un cub de costat 2 centrat a l'origen:

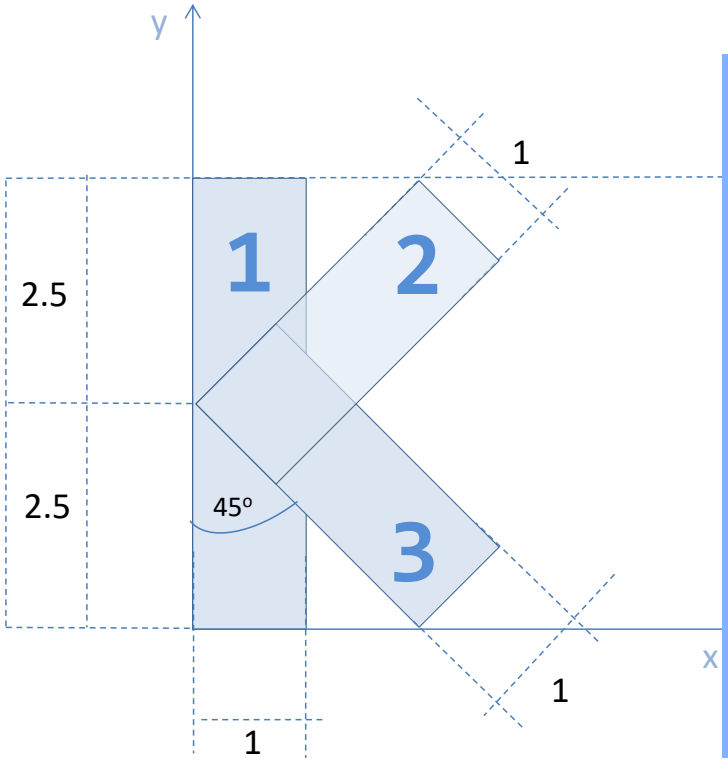


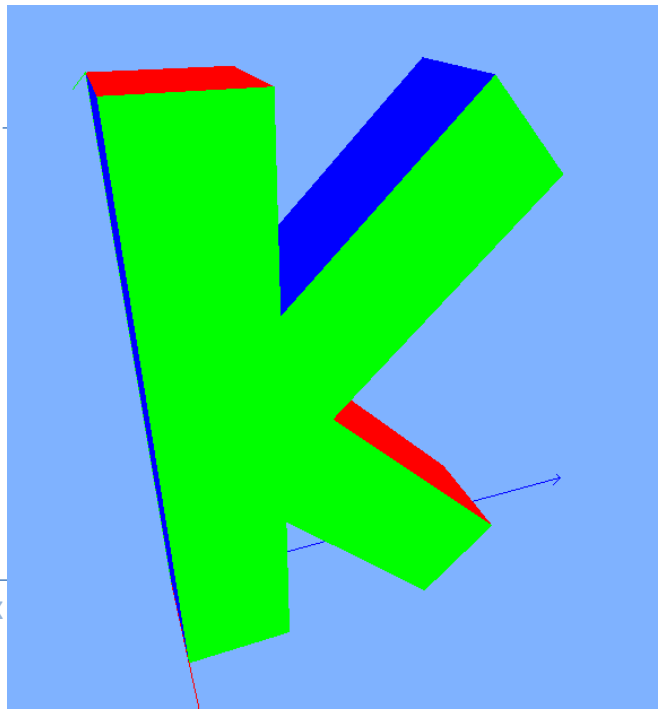
1) [0.5p] Per visualitzar el VAO del cub (sense incloure els eixos!) fem:

```
glBindVertexArray (VAO_Cub);
glDrawArrays(GL_TRIANGLES, 0,  XXXXX   );
```

Què hem de posar a **XXXXX** ? Justifica la resposta.

2) [4p] Volem visualitzar aquest caràcter format per cubs transformats, recolzat en el pla z=0, i de profunditat 1 vers les z positives:





Escriu les transformacions geomètriques necessàries per passar del cub original als tres cubs finals usant les funcions de la llibreria *glm*. Fes-ho programant les funcions **`modelTransformCubX()`**, on ***X*** és el número de fragment (1 o 2 o 3 segons la numeració blava a la figura anterior):

```
void modelTransformCubX() {
     glm::mat4 TG(1.0f);
     ...
     glUniformMatrix4fv (transLoc, 1, GL_FALSE, &TG[0][0]);
}
```

IMPORTANT: Deixeu els càlculs indicats, no només el resultat

```
void MyGLWidget::modelTransformK(int f) {
    float h=5;
    float ln = sqrt(2)*2.5;
    float a = M_PI/4.f;
    glm::mat4 transform (1.0f);

    if(f==1) {  // segment 1
        transform = glm::scale(transform, glm::vec3(1,h,1));
        transform = glm::translate(transform,  glm::vec3(0.5,0.5,+0.5));
        transform = glm::scale(transform, glm::vec3(0.5f));
    } else if(f==2) {  // segment 2
        transform = glm::translate(transform,  glm::vec3(0,h/2,0));
        transform = glm::rotate(transform, -a, glm::vec3(0,0,1));
        transform = glm::scale(transform, glm::vec3(1,ln,1));
        transform = glm::translate(transform,  glm::vec3(0.5,0.5,+0.5));
        transform = glm::scale(transform, glm::vec3(0.5f));
    } else { // segment 3
        transform = glm::translate(transform,  glm::vec3(0,h/2,0));
        transform = glm::rotate(transform, -a, glm::vec3(0,0,1));
        transform = glm::scale(transform, glm::vec3(ln,1,1));
        transform = glm::translate(transform,  glm::vec3(0.5,0.5,+0.5));
        transform = glm::scale(transform, glm::vec3(0.5f));
    }

    glUniformMatrix4fv(transLoc, 1, GL_FALSE, &transform[0][0]);
}
```

3) [4p] Volem crear una càmera que ens proporcioni una vista en perspectiva, sobre el caràcter de l'exercici 2), assegurant una visibilitat total del caràcter **des de qualsevol direcció** de visualització i assumint que
   a. La distancia a la càmera és fixa : 10.
   b. La càmera sempre apunta al centre del caràcter.
   c. El Widget és mostra amb una resolució de 300x1200*px*.

Es demana que calculeu els paràmetres de la matriu de projecció:

```
glm::mat4 Proj = glm::perspective(FOV, ra, znear, zfar);
```

IMPORTANT: Deixeu els càlculs indicats, no només el resultat

Bounding box

$$V_{min} = (0,0,0)$$

$$V_{max} = (\,2.5 + \frac{\sqrt{2}}{2}, 5, 1)$$

$$R = \frac{|V_{max} - V_{min}|}{2} = 0.5 * (\sqrt{\left(2.5 + \frac{\sqrt{2}}{2}\right)^2 + 5^2 + 1^2} = 3.012$$

zNear = 10 -R = 6.988
zFar = 10 +R = 13.012

α_v= asin(R/d) = asin(3.012/10) = 0.3059 rad = 17.52 graus

ra_v = 300/1200 = 0.25
Com que ra_w = 1 > ra_v, cal reduir ra_w → ampliar l'açada de W.

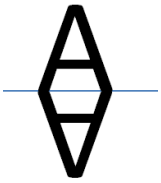
Recalculem la nova apertura vertical segons ra_v:

FOV* = 2* atan(tg(α_v) / ra_v)
FOV* = 1.751669 rad = 103.27 graus

ra_w* = 300/1200 = 0.25

```
glm::mat4 Proj = glm::perspective(1.751669f, 0.25f, 6.988f, 13.012f);
```

4) [1.5p] Volem dibuixar un reflex vertical (simetria respecte l'eix X) del caràcter del punt 2). Se'ns dona la funció **`dibuixaCaracter()`** que fa tota la feina de crear les transformacions (TG,PM i VM) i enviar els vèrtex per dibuixar el caràcter "normal" al vèrtex shader. Sabent que no podem modificar la funció **`dibuixaCaracter()`**, indica que modificaries/afegiries als codis següents per aconseguir dibuixar el caràcter i el seu reflex:



MyGlWidget.cpp

```
void MyGLWidget::paintGL () {
...
   dibuixaCaracter() // dibuixa el caràcter
   dibuixaCaracter() ; // dibuixa el reflex
...
}
```

shader.vert

```
in vec3 vertex;
uniform mat4 TG;
uniform mat4 PM;
uniform mat4 VM;
void main()  {
    gl_Position = PM * VM * TG * vec4 (vertex, 1.0);
}
```

Assumint que la variable "INV" del shader està registrada com a INVLoc:

MyGlWidget.cpp

```
void MyGLWidget::paintGL () {
...
glUniform1f(INVLoc, 1);
   dibuixaCaracter() // dibuixa el caràcter

glUniform1f(INVLoc, -1);
   dibuixaCaracter() ; // dibuixa el reflex
...
}
```

shader.vert

```
in vec3 vertex;
uniform mat4 TG;
uniform mat4 PM;
uniform mat4 VM;
uniform float INV;

void main()  {
    gl_Position = PM * VM * TG * vec4 (vertex.x, vertex.y*INV ,vertex.z,
1.0);
}
```