(18)Una escena està formada per dos cubs, un de costat 20 centrat al punt (0,0,0), i l'altre de costat 10 centrat al punt (15,0,0). Indiqueu TOTS els paràmetres d'una càmera que permeti veure a la vista dos quadrats, un damunt de l'altre (el més gran a sota), de manera que ocupin el màxim de la vista (*viewport*). Cal que indiqueu la posició i orientació de la càmera especificant **VRP**, **OBS** i **up**.

•

47. Considera el següent fragment de codi que defineix una càmera perspectiva:

```
1. glMatrixMode(GL_PROJECTION);
2. glLoadIdentity();
3. gluPerspective(60, 1, zNear, zFar);
4. glMatrixMode(GL_MODELVIEW);
5. glLoadIdentity();
6. glTranslate(0, 0, -10);
7. glRotate( 90,0,0,1);
8. glRotate(-90, 0, 1, 0);
```

Indica i justifica els valors dels paràmetres **OBS, VRP i up que hauries d'utilitzar per a definir la mateixa càmera amb gluLookAt(...).**

58. Indica quina de les inicialitzacions de l'òptica perspectiva és més apropiada per a una càmera que porta un observador que camina per una escena fent fotos. Esfera englobant d'escena té radi R, d és la distància entre OBS i VRP. Observació: ra_v és la relació d'aspecte del *viewport*.

a. FOV=60º, ra=ra_v, zNear=0.1, zFar= 20
b. FOV=60º, ra=ra_v, zNear=R, zFar= 3R;
essent R el radi de l'esfera contenidora de l'escena.
c. FOV=2*(arcsin(R/d)*180/PI); ra=ra_v, zNear=R, zFar= 3R;
essent R el radi de l'esfera contenidora de l'escena i d la distància d'**OBS a VRP.**
d. FOV=2*(arcsin(R/d)*180/PI), ra=ra_v, zNear=0, zFar= 20;
essent R el radi de l'esfera contenidora de l'escena i d la distància d'**OBS a VRP.**

- Una esfera de radi 1 es visualitza en un viewport quadrat de 400 per 400, amb una càmera posicionada correctament amb angles d'Euler, i on el mètode per a definir la projecció de la càmera utilitza la següent crida:

  gluPerspective(60.0, 1.0, 1.0, 10.0);

  L'usuari ha redimensionat la finestra a 500 d'amplada per 400 d'alçada. Digues què cal canviar de la càmera per tal que es vegi l'esfera correctament (sense retallar-la ni deformar-la).

a. Incrementar l'angle d'obertura vertical (FOV) i la relació d'aspecte del window.
b. Augmentar la relació d'aspecte del window i la distància al ZNear.
c. Només augmentar la relació d'aspecte del window.
d. Només canviar l'angle d'obertura vertical (FOV).

- Quan s'inicialitza la càmera, en quin ordre cal indicar les transformacions de càmera i el *viewport* a OpenGL?

a. No importa l'ordre en què s'indiquen.
b. Transformació de posició+orientació, transformació de projecció, *viewport*.
c. La transformació de projecció, transformació de posició+orientació, *viewport*.
d. *Viewport*, tansformació de projecció, transformació de posició+orientació.

Disposem d'una càmera axonomètrica amb els següents paràmetres: OBS=(0.,0.,0.), VRP=(-1.,0.,0.), up=(0.,1.,0.), *window* de (-5,-5) a (5,5), zn=5, zf=10.
Indiqueu quin altre conjunt de paràmetres de càmera defineix exactament el mateix volum de visió (és a dir, garanteix generar exactament la mateixa imatge de l'escena):

OBS= (1,0,0), VRP= (0,0,0), up=(0,2,0), zn= 6, zf=11
OBS= (0,1,0), VRP=(0,0,0), up= (0,1,0), zn=5, zf=10
OBS= (0,0,0), VRP=(-2,0,0), up=(0,1,0), zn=6, zf=11
OBS= (-1,0,0), VRP=(0,0,0), up=(0,1,0), zn=-1, zf=9

43. Donats els següents metodes pintaTriangle() i paintGL():
Indica (o dibuixa) raonadament que es veuria en una vista de 600x600 píxels en els següents casos:
a) Es fa una crida a pintaTriangle() en el punt (A) del codi del paintGL()
b) Es fa la crida en el punt (B)

```
void pintaTriangle ()
{
glColor (1,0,0);
glBegin (GL_TRIANGLES);
glVertex3f (-1, -1, -dist);
glVertex3f (1, -1, -dist);
glVertex3f (0, 1, -dist);
glEnd();
}
```

```
void paintGL (void)
{
// Esborrem els buffers
glClear (GL_COLOR_BUFFER_BIT |
GL_DEPTH_BUFFER_BIT);
// Parametres de camera
glMatrixMode (GL_PROJECTION);
glLoadIdentity ();
glOrtho (-1, 1, -1, 1, dist-1, dist+1);
glMatrixMode (GL_MODELVIEW);
glLoadIdentity ();
//(A)
glTranslatef (0, 0, -dist);
glRotatef (90, 0.0, 1.0, 0.0);
glTranslatef (0, 0, dist);
// (B)
}
```

35. Un professor demana a dos estudiants que li escriguin el codi OpenGL necessari per a definir una càmera que mostra una certa escena en una vista en planta. Els estudiants responen amb dos codis diferents.
Es demana:
• Compleixen l'objectiu demanat cadascun d'aquests dos codis?
• Defineixen tots dos la mateixa càmera?

Codi 1:

```
1 glMatrixMode ( GL_PROJECTION );
2 glLoadIdentity ();
3 glOrtho ( -100 , 100 , -100, 100 , 10, 150);
4 glMatrixMode ( GL_MODELVIEW );
5 glLoadIdentity ();
6 gluLookAt (0, 80, 0, 0, 50, 0, 1, 0, 0);
```

Codi 2:

```
1 glMatrixMode ( GL_PROJECTION );
2 glLoadIdentity ();
3 glOrtho ( -100 , 100 , -100, 100 , 10, 150);
4 glMatrixMode ( GL_MODELVIEW );
5 glLoadIdentity ();
6 glTranslatef (0, 0, -80);
7 glRotatef (90 , 0, 0, 1);
8 glRotatef (90 , 1, 0, 0);
9 glRotatef (-90, 0, 1, 0);
```

- **4 (1 punt)** Disposem d'una càmera ortogonal amb els següents paràmetres: OBS=(0.,0.,0.), VRP=(-1.,0.,0.), up=(0.,1.,0.), window de (-5,5) a (5,5), ra=1, zn=5, zf=10. Indiqueu quin conjunt de paràmetres d'una càmera perspectiva defineix un volum de visió que conté l'anterior (és a dir, garanteix que es veurà, coma mínim, el mateix que amb la càmera axonomètrica):

1. FOV= 90, ra=1, zn= 5, zf=10
2. FOV= 60, ra=1, zn=5, zf=10
3. FOV= 60, ra= 2, zn=6, zf=11
4. FOV= 90, ra= 0.5, zn=5, zf=10

28. S'està visualitzant un objecte amb una càmera perspectiva. A l'avançar la càmera cap a l'objecte (només es modifica la posició de l'observador), s'obté un efecte semblant al zoom. Passa el mateix amb una càmera axonomètrica? Per què?.

4. Disposem del model geomètric de les cares i vèrtexs d'un cub amb longitud d'aresta 1 i ubicat en una posició i orientació arbitrària en l'espai. L'usuari selecciona una de les seves cares (cara_i).

Descriviu el procediment per a calcular totes les inicialitzacions d'una càmera (posició, orientació, tipus,...) que permeti visualitzar la cara_i en pantalla en *verdadera magnitud*; és a dir, que multiplicant per un mateix factor d'escala les seves mides en píxels és corresponguin a les reals. El viewport és de 1024x1024. Suposeu que teniu accés a tota la informació geomètrica de la cara_i.