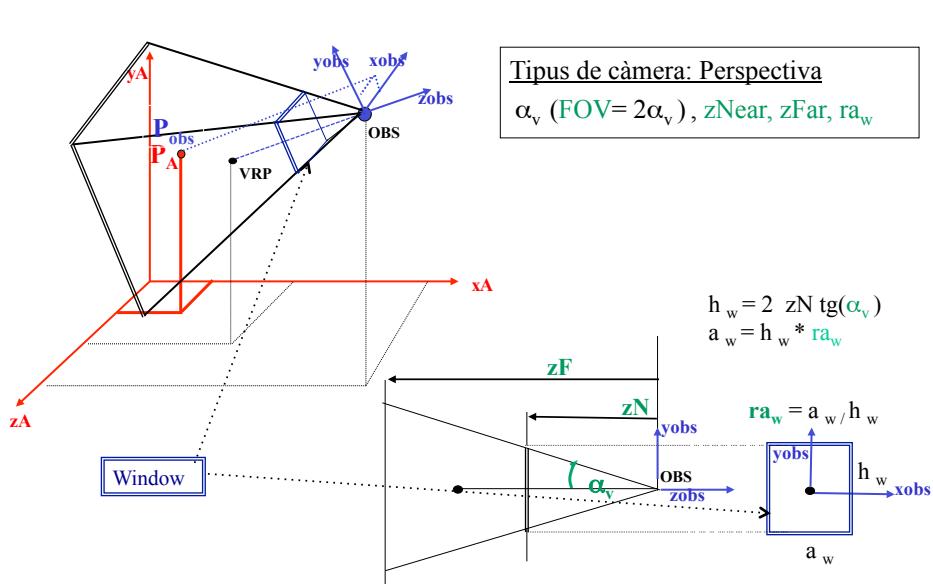


## Classe 5: contingut

- Òptica ortogonal: definició
- Zoom
- Exercicis



## Òptica ortogonal: paràmetres

**Tipus de càmera: ortogonal:**

$l, r, b, t, \text{-window-}, z\text{Near}, z\text{Far}$

$h_w = t - b$   
 $a_w = r - l$   
 $ra_w = a_w / h_w$

(l, b) (r, t)

Window

VRP

OBS

zNear

zFar

yA

xA

zA

yobs

xobs

zobs

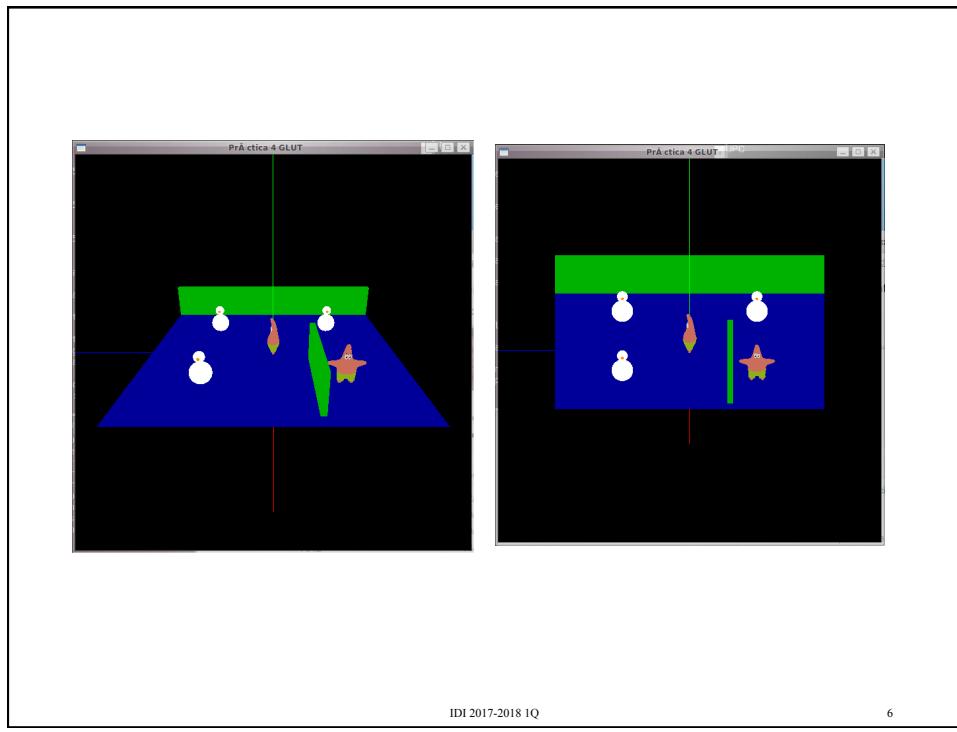
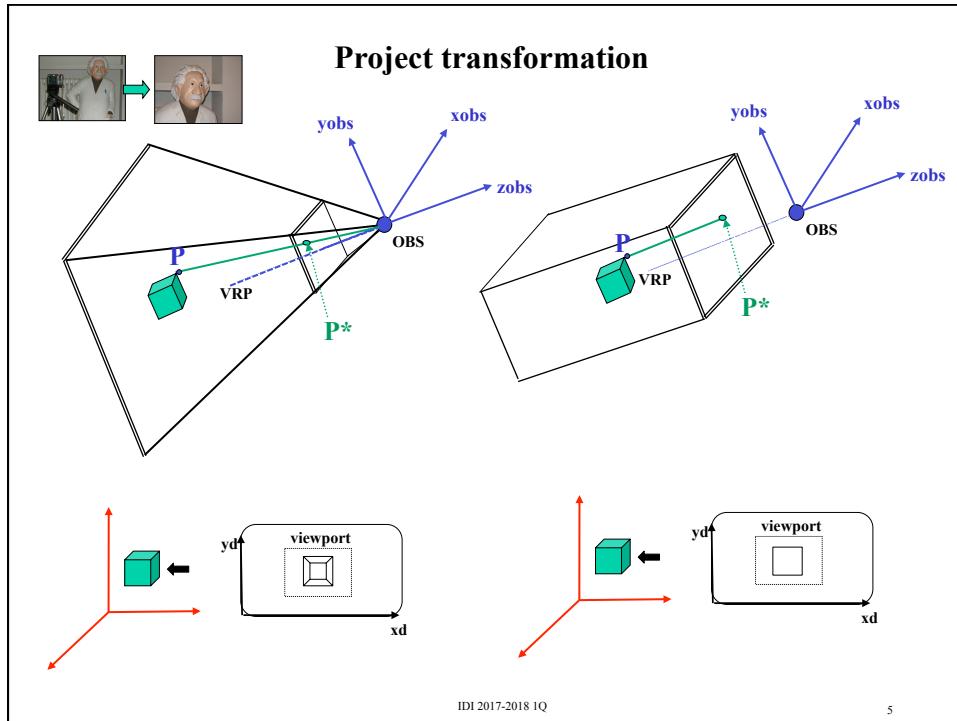
IDL 2017-2018 1Q

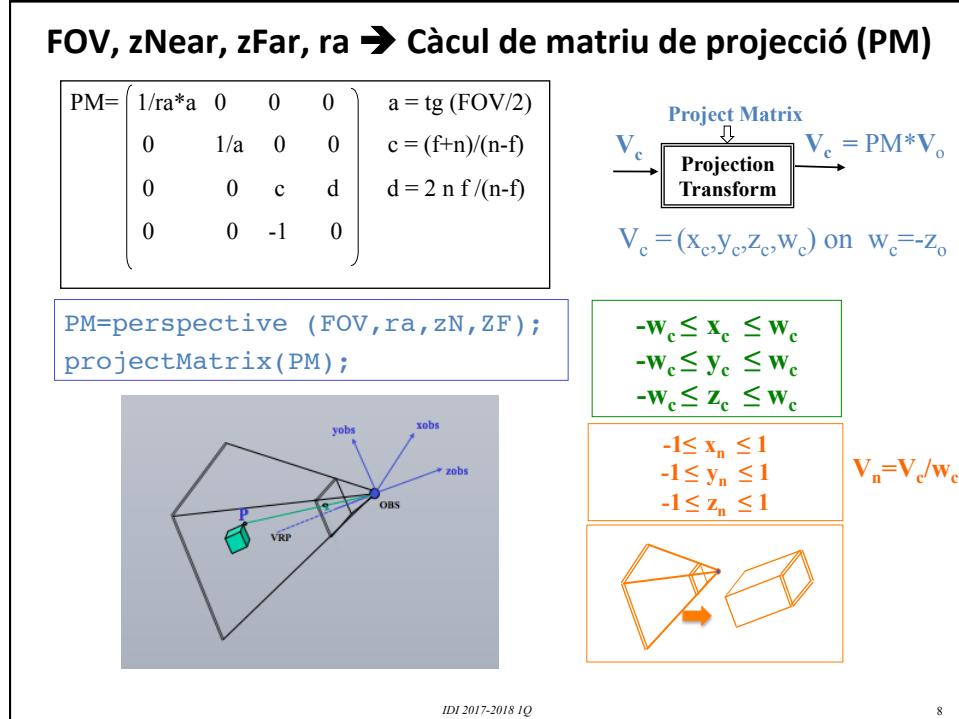
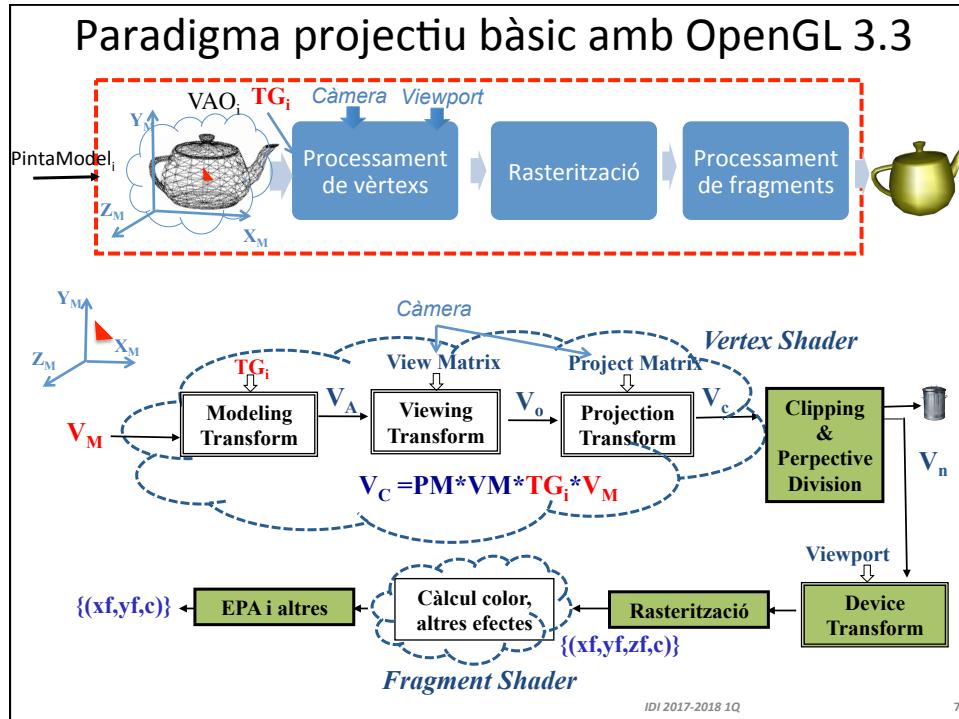
3

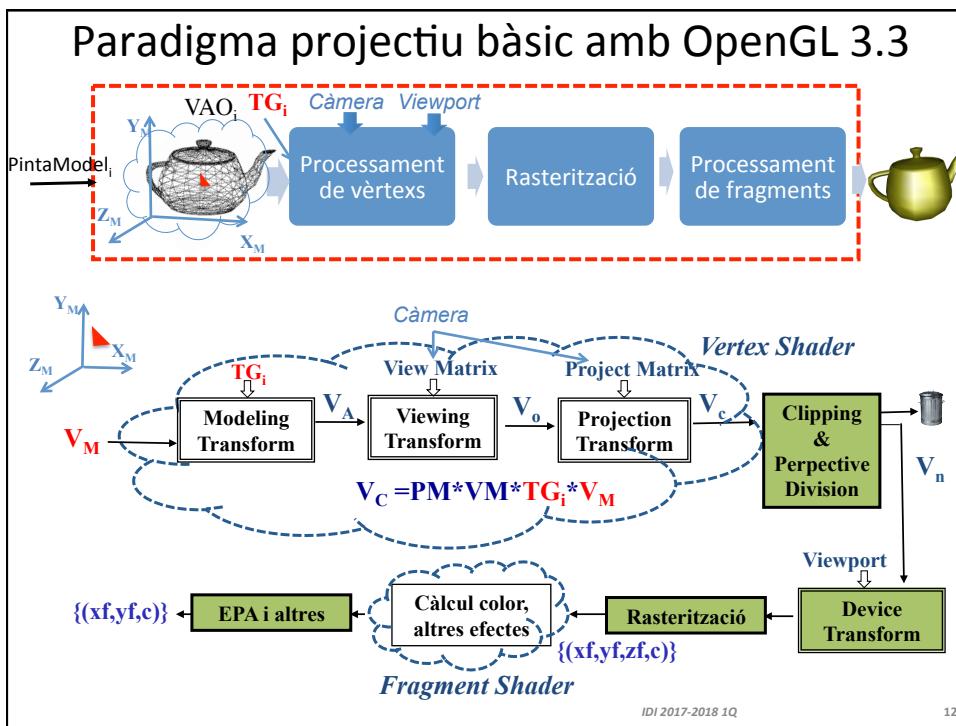
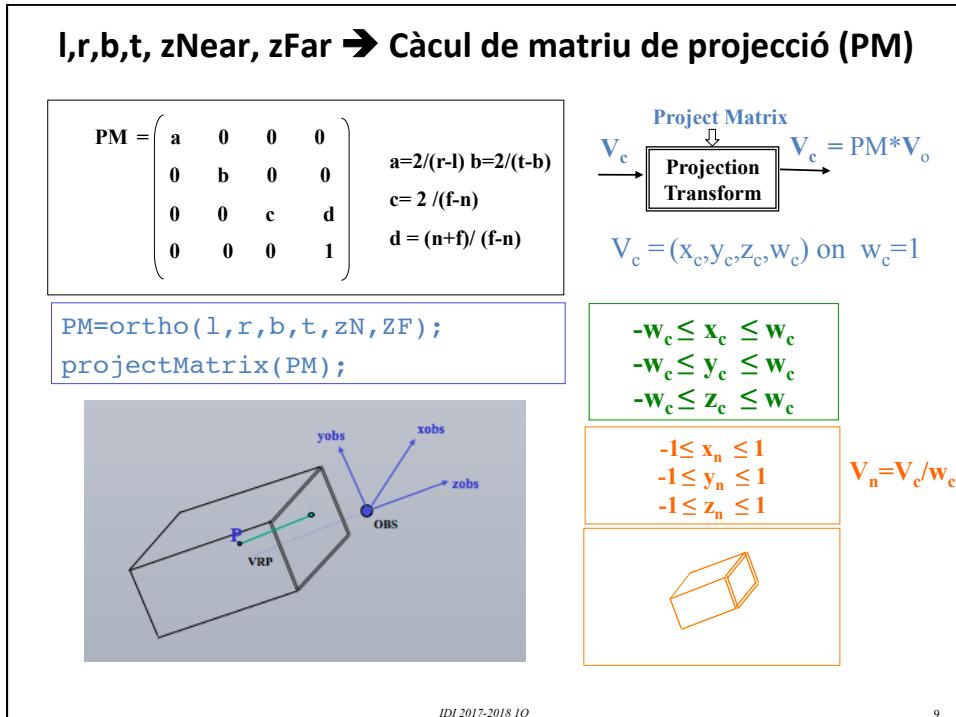
## Òptica ortogonal versus perspectiva

IDL 2017-2018 1Q

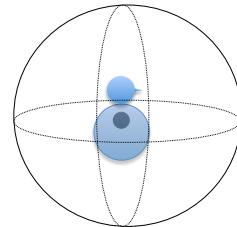
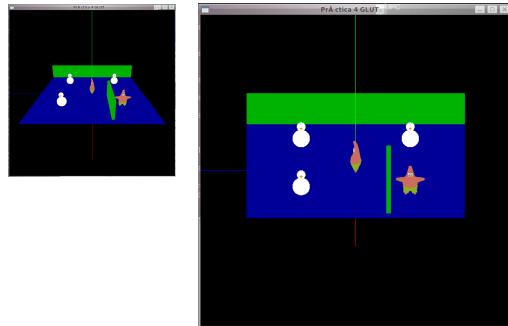
4







## Òptica ortogonal per càmera 3ra persona



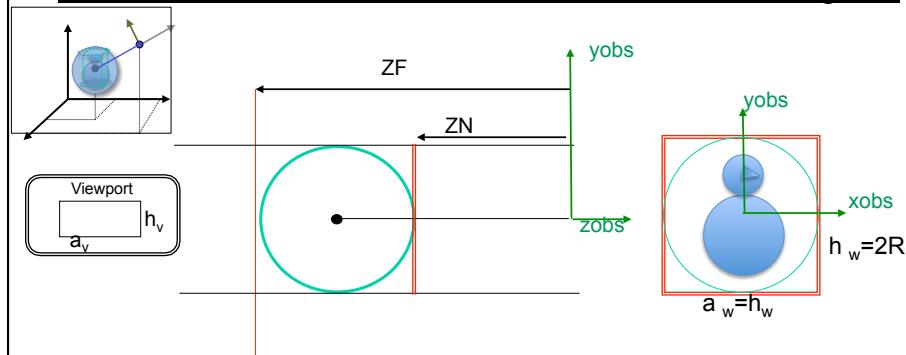
Quins paràmetres de posició, orientació i òptica per càmera en 3ra persona? → imatge inclogui tota l'escena, optimitzant l'espai del viewport.

Dada: capsa mínima contenidora ( $x_{min}, y_{min}, z_{min}$ ) - ( $x_{max}, y_{max}, z_{max}$ )

IDI 2017-2018 1Q

13

### Tota escena en la vista, sense deformar i càmera ortogonal



- $ZN$  i  $ZF$  mateix raonament que en càmera perspectiva.
- Window mínim requerit (centralitzat) =  $(-R, R, -R, R)$  => una  $ra_w = 1$  (per què?)

• Si  $ra_w \neq ra_v \Rightarrow$  deformació (per què?)  
 - Si  $ra_v > 1 \Rightarrow$  cal incrementar la  $ra_w \Rightarrow$  modificar window  
 com  $ra_w = a_w/h_w \Rightarrow$  podem incrementar  $a_w$  o decrementar  $h_w$  (és retallaria esfera!!)  
 Per tant:  
 $a_w^* = ra_v * h_w = ra_v * 2R$   
 $window = (-R ra_v, R ra_v, -R, R)$

- raonament similar per recalcular window quan  $ra_v < 1$

IDI 2017-2018 1Q

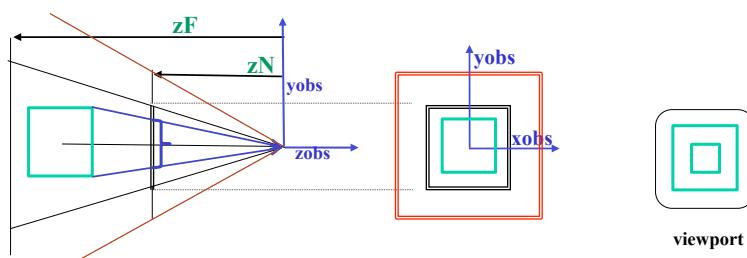
14

## Classe 5: contingut

- Òptica axonomètrica: definició
- Zoom
- Exercicis

## L'òptica i el ZOOM

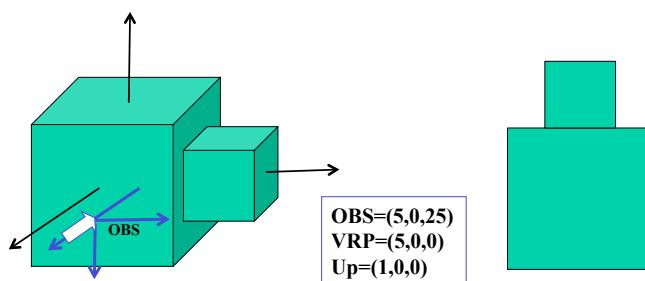
- a) Modificar l'angle d'obertura (tot mantenint la ra) ;  
**Modificar window en ortogonal**
- b) Modificar la distància de l'Obs al VRP  
(modificant ZN i ZF adequadament)
- c) Modificar Obs i VRP en la direcció  $-v$  → travelling



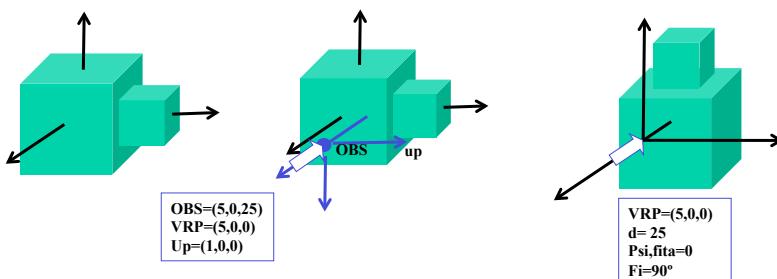
## Classe 5: contingut

- Òptica axonomètrica: definició
- Zoom
- Exercicis

**Exercici 13.** 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; a) **VRP**, **OBS** i **up** b) angles Euler



**Exercici 13(bis).** 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 els **angles d'Euler** i indicant l'expressió de la *viewMatrix*.



IDI 2017-2018 1Q

20

**Exercici 45.** Una esfera de radi 1 es visualitza en un viewport quadrat de 400 per 400, amb una càmera posicionada correctament per poder veure tota l'esfera, i on el mètode per a definir la projecció de la càmera utilitza la següent crida:

```
TP = Perspective (60.0, 1.0, 1.0, 10.0);
projectMatrix (TP);
```

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).

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

IDI 2017-2018 1Q

21

**Exercici:** 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*, transformació de projecció, transformació de posició + orientació.

```
/* CreateBuffers(); Crear VAO del model
   (un cop)*/

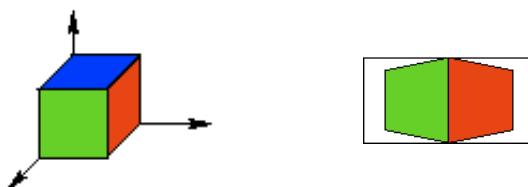
...
/*IniCamera() calcular paràmetres càmera i
   matrius cada cop que es
   modifiquin */
//viewTransform()
VM = lookAt(OBS,VRP,UP);
viewMatrix(VM);
//projectTransform()
PM=perspective (FOV,ra,zN,ZF);
projectMatrix(PM);
//resize(...)
glViewport (0,0,w.h);
/*PaintGL(); cada cop que es requerix
   refresc*/
/*per cada model: modelTransform()
   Calcula TG i passar a OpenGL*/
modelTransform_i(TG);
modelMatrix(TG);
Pinta_model(VAO);
```

### Vertex Shader

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

77. (2015-2016P Q1) Tenim una escena amb un cub de costat 2 orientat amb els eixos i de manera que el seu vèrtex mínim està situat a l'origen de coordenades. La cara del cub que queda sobre el pla  $x=2$  és de color vermell, la cara que queda sobre el pla  $z=2$  és de color verd i la resta de cares són blaves.

a) Indica TOTS els paràmetres d'una càmera perspectiva que permeti veure completes a la vista només les cares vermella i verda. La relació d'aspecte del viewport (vista) és 2. Fes un dibuix indicant la imatge final que s'obtindria.



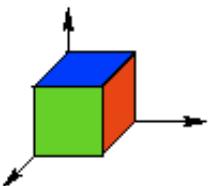
IDI 2017-2018 IQ

25

77. (2015-2016P Q1) Tenim una escena amb un cub de costat 2 orientat amb els eixos i de manera que el seu vèrtex mínim està situat a l'origen de coordenades. La cara del cub que queda sobre el pla  $x=2$  és de color vermell, la cara que queda sobre el pla  $z=2$  és de color verd i la resta de cares són blaves.

a) Indica TOTS els paràmetres d'una càmera perspectiva que permeti veure completes a la vista només les cares vermella i verda. La relació d'aspecte del viewport (vista) és 2. Fes un dibuix indicant la imatge final que s'obtindria.

b) Quin efecte tindria en la imatge final modificar l'òptica a axonomètrica?



IDI 2017-2018 IQ

27

**Exercici 43:** 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 amb una òptica constant. 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)  $\text{FOV} = 60^\circ$ ,  $ra = ra\_v$ ,  $zNear = 0.1$ ,  $zFar = 20$
- b)  $\text{FOV} = 60^\circ$ ,  $ra = ra\_v$ ,  $zNear = R$ ,  $zFar = 3R$ ;  
essent  $R$  el radi de l'esfera contenidora de l'escena.
- c)  $\text{FOV} = 2 * (\arcsin(R/d) * 180/\pi)$ ,  $ra = ra\_v$ ,  $zNear = R$ ,  $zFar = 3R$ ;  
essent  $R$  el radi de l'esfera contenidora de l'escena i  $d$  la distància d'OBS a VRP.
- d)  $\text{FOV} = 2 * (\arcsin(R/d) * 180/\pi)$ ,  $ra = ra\_v$ ,  $zNear = 0$ ,  $zFar = 20$ ;  
essent  $R$  el radi de l'esfera contenidora de l'escena i  $d$  la distància d'OBS a VRP

73. (2014-2015P 2Q) Es vol realitzar una vista en planta (visió des de dalt) d'una escena/objecte que està centrat a l'origen amb una capsa contenidora de mides  $10 \times 10 \times 10$ . Quina de les següents definicions et sembla correcta per definir la posició + orientació de la càmera (per a calcular la viewMatrix)? Sabem que la càmera és perspectiva i els angles de les rotacions estan en graus.

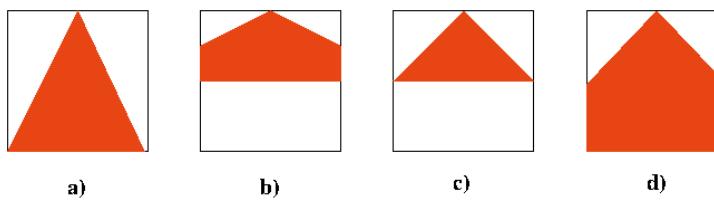
- a)  $\text{OBS} = (0, 10, 0)$ ;  $\text{VRP} = (0, 0, 0)$ ;  $\text{up} = (0, 1, 0)$ ;  
 $\text{VM} = \text{lookAt}(\text{OBS}, \text{VRP}, \text{up})$ ;  
 $\text{viewMatrix}(\text{VM})$ ;
- b)  $\text{OBS} = (0, 0, 0)$ ;  $\text{VRP} = (0, 10, 0)$ ;  $\text{up} = (0, 0, -1)$ ;  
 $\text{VM} = \text{lookAt}(\text{OBS}, \text{VRP}, \text{up})$ ;  
 $\text{viewMatrix}(\text{VM})$ ;
- c)  $\text{VM} = \text{translate}(0, 0, -10)$ ;  
 $\text{VM} = \text{VM} * \text{rotate}(90, 1, 0, 0)$ ;  
 $\text{viewMatrix}(\text{VM})$ ;
- d)  $\text{VM} = \text{translate}(0, 0, -10)$ ;  
 $\text{VM} = \text{VM} * \text{rotate}(-90, 0, 1, 0)$ ;  
 $\text{viewMatrix}(\text{VM})$ ;

1. (2014-2015P 2Q) Cal definir una càmera a OpenGL; quin dels següents pseudocodis és correcte? Noteu que tant sols canvia l'ordre en què es fan les crides.

1) VM=lookAt(OBS, VRP, up) viewMatrix (VM) PM=perspective (FOV, ra, zn,zf) projectMatrix(PM) glViewport(...) modelMatrix(TG) pintaescena()	2) modelMatrix(TG) PM=perspective (FOV, ra, zn,zf) projectMatrix(PM) VM=lookAt(OBS, VRP, up) viewMatrix (VM) glViewport(...) pintaescena()
3) VM=lookAt(OBS, VRP, up) viewMatrix (VM) PM=perspective (FOV, ra, zn,zf) projectMatrix(PM) modelMatrix(TG) glViewport(...) pintaescena()	4) glViewport(...) VM=lookAt(OBS, VRP, up) viewMatrix (VM) PM=perspective (FOV, ra, zn,zf) projectMatrix(PM) modelMatrix(TG) pintaescena()

- a) només 1) i 4) són correctes
- b) només 4) és correcte
- c) tots són correctes
- d) tots són correctes menys 2)

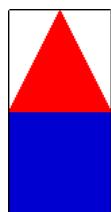
Tenim una escena amb un triangle vermell amb vèrtexs  $V1=(-2,0,0)$ ,  $V2 = (2, 0, 0)$  i  $V3=(0, 1, 0)$ . Suposant que tenim un viewport quadrat de 600x600 pixels, i que hem inicialitzat les matrius de càmera (view) i projecció (proj) a la matriu identitat, indica quina de les següents imatges és la que sortirà en un viewport de 600x600 (sabem que el Vertex Shader i el Fragment Shader estan correctament implementats):



100. (2016-2017P Q1) Dos estudiants discuteixen respecte a la implementació del zoom amb àptica axonomètrica (ortogonal) i perspectiva. Quina de les seves afirmacions és certa?

- a) En àptica ortogonal només es pot obtenir un efecte de zoom modificant OBS i VRP en la direcció de visió.
- b) En àptica perspectiva cal modificar FOV, Znear i Zfar.
- c) En les dues àptiques es pot fer zoom modificant el window de la càmera.
- d) En àptica perspectiva si avancem OBS i VRP en la direcció de visió cal anar amb compte amb la ra.

Tenim una piràmide de base quadrada de costat 5, amb la base centrada al punt (0,0,2.5) i alçada de la piràmide 5 amb l'eix en direcció Z+. A l'escena tenim també un cub de costat 5 centrat a l'origen. El viewport esta definit amb glViewport (0,0,400,800). Si a la vista es veu la imatge que tenui al dibuix (caseta), quines inicialitzacions d'una càmera axonomètrica (posició+orientació i àptica) permetrien veure aquesta imatge? Tots els angles estan en graus.



<pre> PM=perspective (90, 1, 5, 10); projectionMatrix (PM) VM=translate (0,0,-10); VM=VM*rotate (90,1,0,0); VM=VM*translate (0,0,-2.5); viewMatrix (VM); pinta_escena (); </pre>	<pre> PM=ortho (-2.5, 2.5, -5, 5, 5, 10); projectionMatrix (PM) VM=translate (0,0,-7.5); VM=VM*rotate (-90,0,0,1); VM=VM*rotate (90,0,1,0); VM=VM*translate (0,0,-2.5); viewMatrix (VM); pinta_escena (); </pre>
<pre> PM=ortho (-2.5, 2.5, -5, 5, 5, 10); projectionMatrix (PM) VM=translate (0,0,-7.5); VM=VM*rotate (90,0,0,1); VM=VM*rotate (90,0,1,0); VM=VM*translate (0,0,-2.5); viewMatrix (VM); pinta_escena (); </pre>	<pre> PM=ortho (-5, 5, -5, 5, 5, 10); projectionMatrix (PM) VM=translate (0,0,-7.5); VM=VM*rotate (90,0,0,1); VM=VM*rotate (90,0,1,0); VM=VM*translate (0,0,-2.5); viewMatrix (VM); pinta_escena (); </pre>

**Exercici 48:** 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):

- a) FOV= 90, ra=1, zn= 5, zf=10
- b) FOV= 60, ra=1, zn=5, zf=10
- c) FOV= 60, ra= 2, zn=6, zf=11
- d) FOV= 90, ra= 0.5, zn=5, zf=10