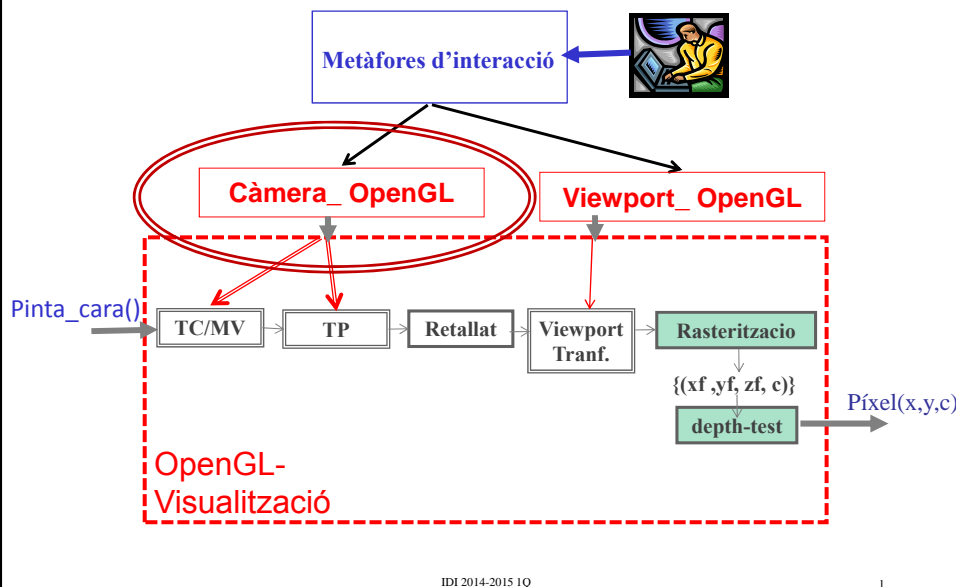
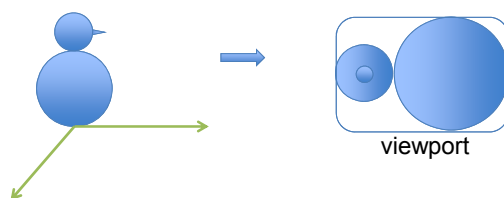


## Visualització OpenGL: Pipeline bàsic



**Exemple 1:** Donada una funció `pinta_ninot()` que pinta l'objecte de la figura, format per: una esfera de radi 10 i centre (0,10,0), altre esfera de radi 5 i centre (0,25,0), i un con de base centrada en (2.5, 25,0),  $r=2$  i llargada 5 orientat segons l'eix X; indicar tots els paràmetres d'una càmera *ortogonal* que permeti obtenir la imatge que s'indica en un viewport de 600x400.



**Cal definir la posició i orientació de la càmera i l'òptica**

**Pinta\_Ninot()**

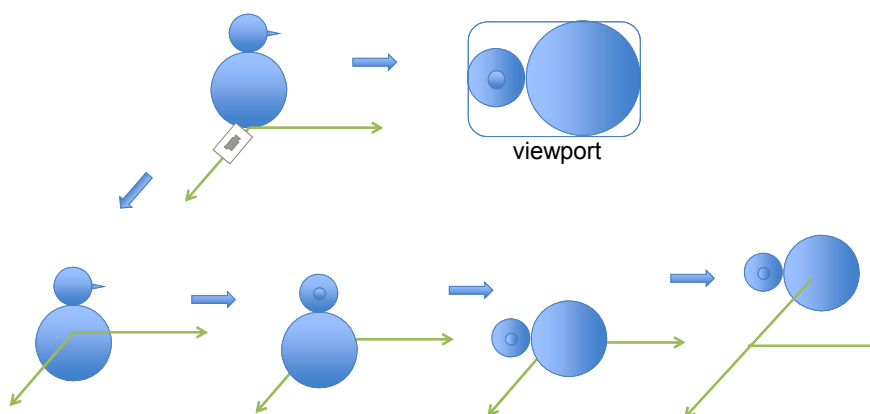
```

glMatrixMode(GL_MODELVIEW);
// esfera 1
glPushMatrix();
glTranslatef (0.,10.,0.);
glutSolidSphere(10.,20,20);
glPopMatrix();
//esfera 2
glPushMatrix();
glTranslatef (0.,25.,0.);
glutSolidSphere(5.,20,20);
glPopMatrix();
//con
glPushMatrix();
glTranslatef(2.5,25,0.);
glRotatef(90.,0.,1.,0.);
glutSolidCone(2.,5.,20,20)
glPopMatrix();

```

IDI 2014-2015 1Q

3

**Exemple 1: Càmera amb TG**

$$TC = T(0,0,-30)G_z(90)G_y(-90)T(0,-15,0)$$

```

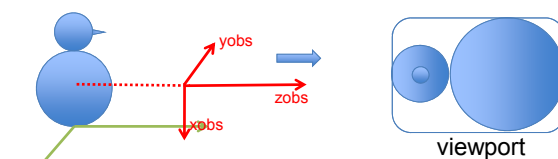
glMatrixMode (GL_MODELVIEW);
glLoadIdentity();
glTranslatef (0,0,-30.);
glRotatef (90.,0.,0.,1.);
glRotatef (-90.,0.,1.,0.);
glTranslatef (0,-15.,0.);
Pinta_Ninot();

```

IDI 2014-2015 1Q

4

### Exemple 1: Càmera amb gluLookAt



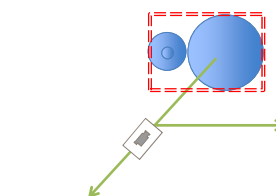
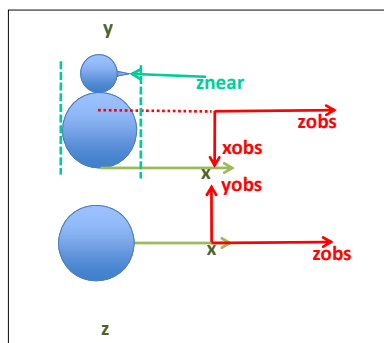
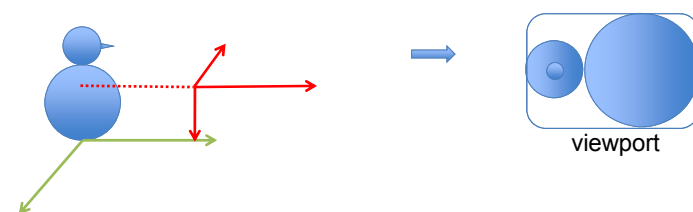
↓ Pensant en càmera i com  
ha de quedar la imatge

VRP=(0,15,0); OBS=(30,15,0), up=(0,0,-1)

IDI 2014-2015 1Q

5

### Exemple 1: Òptica axonomètrica



```
glViewport (0,0,600,400);
glMatrixMode (GL_PROJECTION);
glLoadIdentity();
glOrtho(-15,15,-10,10,20,40);
```

IDI 2014-2015 1Q

6

glMatrixMode (GL\_MODELVIEW);  
glLoadIdentity();  
glTranslatef (0.,0.,-30.)  
glRotatef (90.,0.,0.,1.)  
glRotatef (-90.,0.,1.,0.)  
glTranslatef (0.,-15.,0.)

*Interpretació alternativa de TG per a determinar OBS, VRP i up*

$TC = T(0,0,-30)G_z(90)G_y(-90)T(0,-15,0)$

$TG_c \Rightarrow 1) T(0,15,0) \ 2) G_y(+90) \ 3) G_z(-90) \ 4) T(0,0,30)$

IDI 2014-2015 1Q 7

## Per pensar...

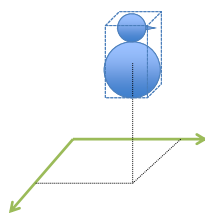
Quins podrien ser uns **paràmetres de posició, orientació i òptica** per a una càmera que, donada una escena amb caps mínima contenidora

(xmin, ymin, zmin) - (xmax, ymax, zmax),

visualitzi una imatge que inclogui totalment l'escena, ocupant el màxim de la vista (viewport) i sense deformació?

### Exercici d'inicialització i moure càmera

- Veure escena sempre sense retallar i sense deformació (en pas a viewport)
- La imatge inicial volem que estigui centrada i ocupant raonablement el viewport i mostri escena vista des d'una posició arbitrària.
- Càmera perspectiva.
- Permetre modificació interactiva de punt de vista.

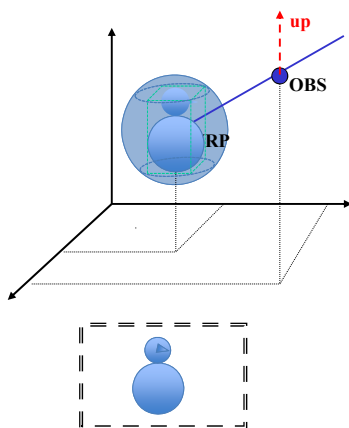


Imaginem aquesta escena

IDI 2014-2015 1Q

9

### Exercici d'inicialització posicionament amb OBS, VRP, up



- Centrat => VRP=CentreEscena

• Per assegurar que escena es veu sense retallar des d'una posició arbitrària CAL que OBS sempre fora capsa mínima contenidora en una posició qualsevol; per assegurar-ho CAL que OBS fora de l'esfera englobant de la capsa => distància "d" de l'OBS a VRP superior a R esfera.

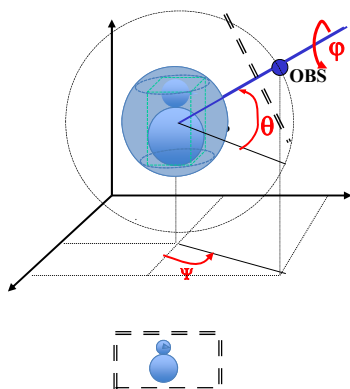
CapsaMinCont=(xmin,ymin,zmin,xmax,ymax,zmax)  
 CentreEscena=Centre(CapsaMinCont) => ((xmax+xmin)/2, ...)  
 $R = \text{dist}((x_{min}, y_{min}, z_{min}), (x_{max}, y_{max}, z_{max})) / 2$   
 $d > R$ ; per exemple  $d = 2R$   
 $OBS = VRP + d * v$ ; v normalitzat en qualsevol direcció;  
 per exemple  $v = (1,1,1) / \|(1,1,1)\|$

- up qualsevol que no sigui paral·lel a v; si volem ninot vertical (eix Y és vegi vertical)  $up = (0,1,0)$

IDI 2014-2015 1Q

10

### Exercici d'inicialització càmera: Posicionament amb angles Euler (TG)



- Imaginem movem la càmera (OBS) sobre una esfera centrada en VRP de radi  $d$ .
  - VRP i  $d$  calculats com s'ha vist en exercici anterior.
  - Podem definir la posició amb angles Euler:  $\psi$  i  $\theta$ .  
per exemple:  $\psi = 45^\circ$  i  $\theta = 45^\circ$  (o altres valors)
  - gir càmera sobre si mateixa:  $\phi$   
per exemple:  $0^\circ$  per veure eix Y vertical.
- Podem indicar càmera a OpenGL amb TG a objecte (més directe) o calculant VRP, OBS i up (veure transpa següent). Recordeu que transformacions "en codi" són als objectes (repasseu exercicis anteriors)

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glTranslatef(0,0,-d)
glRotatef(θ,1,0,0.)
glRotatef(-ψ,0,1,0.)
glTranslatef(-VRP.x,-VRP.y,-VRP.z))
```

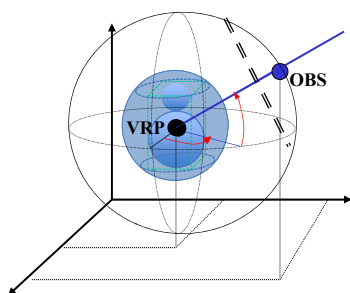
Ull amb signes:

- Si s'ha calculat  $\psi$  positiu quan càmera gira cap a la dreta, serà un gir anti-horari respecte eix Y de la càmera, per tant, matemàticament positiu; com girem els objectes en sentit contrari, cal posar  $-\psi$  en el codi.
- Si s'ha calculat  $\theta$  positiu quan pugem la càmera, serà un gir horari; per tant, matemàticament un gir negatiu; com objecte girarà en sentit contrari (anti-horari), ja és correcte deixar signe positiu.

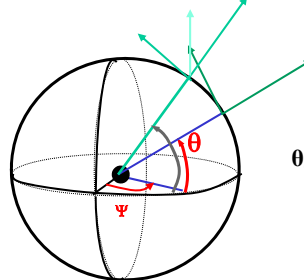
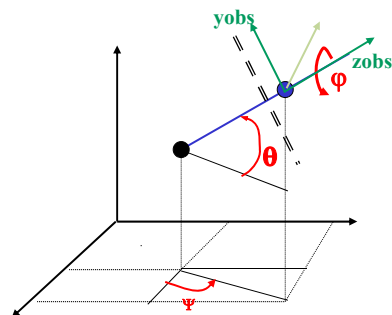
IDI 2014-2015 1Q

11

### Moure la Càmera amb angles d'Euler



- Imaginem que en la interfície d'usuari estem ubicant la Càmera movent el cursor dreta/esquerra ( $\psi$ ) i pujar/baixar ( $\theta$ ). És com moure OBS sobre l'esfera i els angles d'Euler determinen un punt en esfera.
- També ho podeu pensar com si girem l'esfera per a què la seva posició respecte la càmera de defecte sigui la mateixa (agafar l'esfera amb la ma i girar-la).
- Codi per OpenGL directe a partir dels angles.

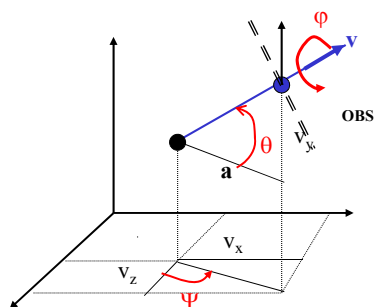


$$\theta_c = \theta_c + \Delta y_{mov}$$

IDI 2014-2015 1Q

12

### Càlcul VRP, OBS a partir angles Euler



**VRP** = Punt d'enfoc

**OBS** = VRP + d **v**

**d** > **R** ; per exemple: d = 2R

$$v_y = \sin(\theta); a = \cos(\theta);$$

$$v_z = \cos(\theta) \cos(\Psi);$$

$$v_x = \cos(\theta) \sin(\Psi);$$

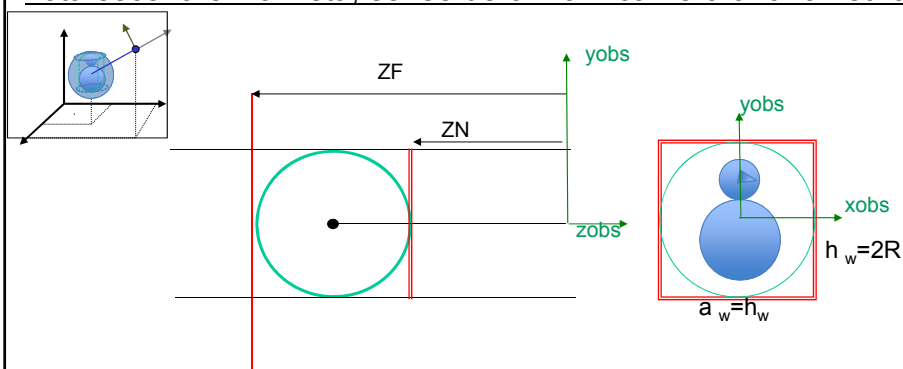
Un possible **up**: **up** = (0,1,0) ( $\phi = 0^\circ$ )

Noteu que "aquí" estem considerant els angles d'orientació de la càmera ==>  $\Psi, \theta$  positius quan movem la càmera cap a la dreta i quan la pugem.

13

IDI 2014-2015 1Q

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

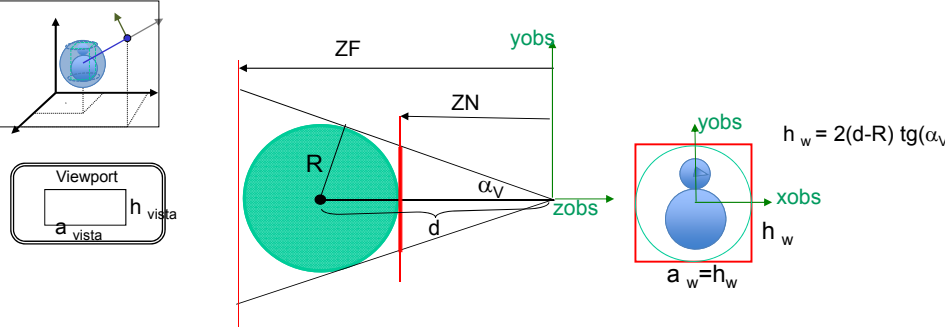


- **ZN i ZF** mateix raonament que en càmera perspectiva.
- **Window mínim requerit (centrat)** = (-R,-R,R,R) => una  $ra_w = 1$  (per què ?)
- Si  $ra_w \neq ra_v$  ==> deformació (per què?)
  - Si  $ra_v > 1$  => cal incrementar la  $ra_w$  => *modificar window*
  - com  $ra_w = a_w/h_w$  => podem incrementar  $a_w$  o decrementar  $h_w$  (és retallar la esfera!!)
  - Per tant:
  - $a_w^* = ra_v * h_w = ra_v * 2R$  =>  $inc\_a = a_w^* - a_w$
  - window = (- (R+inc\_a/2), R+inc\_a/2, -R, R) = (-R  $ra_v$ , R  $ra_v$ , -R, R)
  - raonament similar per recalculer window quan  $ra_v < 1$

14

IDI 2014-2015 1Q

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



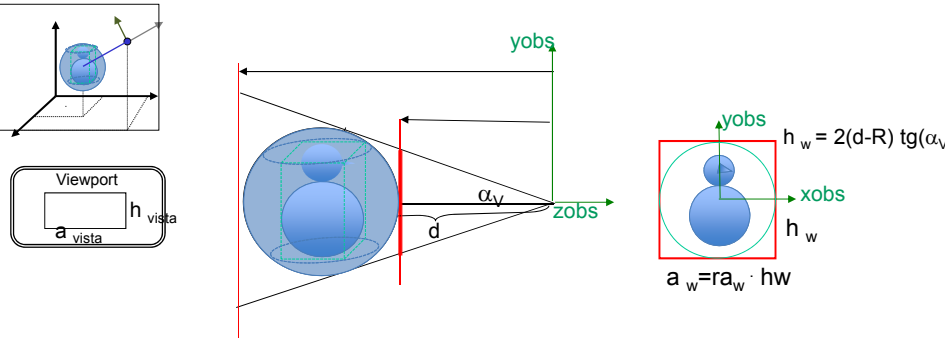
Si tota l'esfera englobant està dins la profunditat del camp de visió, no retallem l'escena.  
Per tant,  $ZN \in ]0, d-R]$   $ZF \in ]d+R, \dots]$ ;  
per a aprofitar precisió profunditat:  $ZN = d-R$ ;  $ZF = d+R$

- Per a aprofitar al màxim la pantalla (de fet el viewport), el window de la càmera s'ha d'ajustar a l'escena; una aproximació és ajustar el volum de visió (piràmide) de la càmera a l'esfera englobant.
  - $R = d \sin(\alpha_v)$  ;  $\alpha_v = \arcsin(R/d)$   $\Rightarrow FOV = 2 * \alpha_v$
  - com window està situat en  $ZN$ ,  $\alpha_v$  determina que la seva alçada sigui:  $h_w = 2(d-R) \operatorname{tg}(\alpha_v)$
- $ra_w = a_w/h_w = 1$  (perquè  $\alpha_H$  hauria de ser igual a  $\alpha_v$  per assegurar que esfera no retallada)
  - Però ULL!! per a què no hi hagi deformació, cal que  $ra_w$  sigui sempre igual a  $ra_v$ , per tant, si no volem modificar el viewport:

$ra_w^* = ra_v$  amb aquesta nova  $ra_w^*$  es retallarà l'esfera? (continuarà tota dins volum de visió?)

15

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



Si  $ra_v > 1$  ( $> ra_w$  mínima requerida 1)  $\Rightarrow$  No es retalla,  
no cal modificar  $\alpha_v$  (FOV)

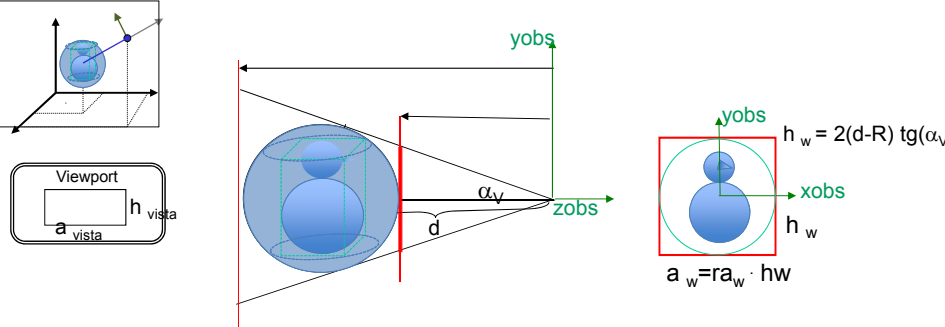
Amb  $ra_w = 1$  viewport

Amb  $ra_w = ra_v$

16



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



• Si  $ra_v < 1$  ( $< ra_w$  mínim requerida 1) => cal incrementar l'angle d'obertura (quedarà espai lliure a dalt i a baix)

$FOV = 2 \alpha_v^*$  on  $\alpha_v^* = \arctg(tg(\alpha_v) / ra_v)$

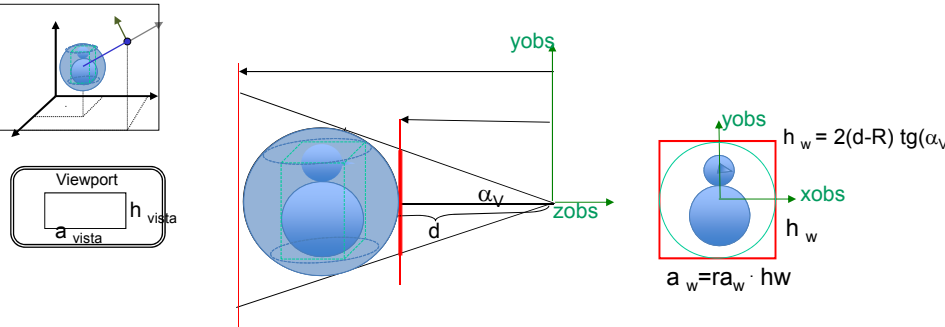
Amb  $ra_w = rav$

• Amb  $ra_w = ra_v$  i nou FOV

• Sempre cal calcular el nou angle a partir de l'inicial (window quadrat). Penseu que passaria si no ho feu i modifiqueu interactivament el viewport (finestra gràfica) fent-ho  $> 1$  i  $< 1$  molts cops seguits.

17

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



• Si  $ra_v > 1$  ( $> ra_w$  mínima requerida 1) => No es retalla, **no cal modificar  $\alpha_v$  (FOV)**

Justificació:  $ra_w^*$  serà superior a 1; si no modifiquem l'angle FOV,  $h_w$  no canvia =>  $a_w^* = ra_w^* \cdot h_w$  i com  $ra_w^* > ra_w$  =>  $a_w^* > a_w$  i, per tant, serà més gran del necessari però es veurà tota l'esfera i quedarà espai pels laterals.

• Si  $ra_v < 1$  ( $< ra_w$  mínim requerida 1) => cal incrementar l'angle d'obertura (quedarà espai lliure a dalt i a baix)

$FOV = 2 \alpha_v^*$  on  $\alpha_v^* = \arctg(tg(\alpha_v) / ra_v)$

Justificació: com  $a_w^* = ra_w^* \cdot h_w$ , si no modifiquem angle,  $h_w$  no varia; com  $ra_w^* < ra_w$  =>  $a_w^* < a_w$  i l'esfera quedaria retallada (en horitzontal). Per tant, cal incrementar l'angle  $\alpha_v$  (i, per tant,  $h_w$ ) per a garantir una amplada del window igual a la mínima requerida (igual que la  $h_w$  inicial).

- sabem:  $h_w^* = a_w / ra_v = (2(d-R)tg(\alpha_v)) / ra_v$  i per trigonometria  $h_w^* = 2(d-R) tg(\alpha_v^*)$

- per tant:  $\alpha_v^* = \arctg(tg(\alpha_v) / ra_v)$

18

**Exercicis de la llista a fer (mínims):**

- 35 \*
- 42
- 45 \*
- 22
- 48
- 52
- 70 \*

Els exercicis marcats amb \* els farem a classe (en la mesura del possible)